

Развивающее программирование

Решение задач на языке C++



Бесплатное издание

Все права защищены. Никакая часть этой книги не может быть воспроизведена в любой форме без письменного разрешения правообладателей.

Автор книги не несёт ответственности за возможный вред от использования информации, составляющей содержание книги и приложений.

Copyright 2016 Валерий Рубанцев
Лилия Рубанцева

От автора

Принято считать, что язык *C++* очень сложный, поэтому его трудно изучить. И это действительно так. Но на «школьном» уровне сложность *C++* сопоставима с *Питоном* или *паскалем*, поэтому его вполне можно изучать наряду с этими языками программирования.

Как и в любом другом деле, при изучении *C++* нужна систематическая тренировка, под которой можно понимать решение задач на основные элементы языка программирования. Это могут быть и формальные упражнения, и занимательные задачи. Принципиальной разницы между ними нет, за исключением того, что занимательные задачи имеют сюжет, а потому более интересны.

В этой книге собрано несколько десятков занимательных математических задач всех времён и народов, начиная от самых древних задач, записанных на папирусах и глиняных табличках, до конкурсных задач из журнала *Квантик*.

Решая задачи, вы укрепите умения и навыки в применении таких элементов языка *C++*, как:

- Числовые типы данных – *int*, *long*, *long long*, *unsigned long long*, *double*
- Логический тип *bool*
- Арифметические операции
- Простейшие математические функции
- Переменные и константы
- Операторы объявления и присваивания
- Комбинированные операторы присваивания
- Логические операторы и выражения
- Условные логические операторы *if*, *if – else*
- Циклы *for*, *while*, *do - while*
- Функции
- Операторы *return*, *continue*, *break*
- Операции ввода и вывода *cin*, *cout*

Все задачи решены в интегрированной среде разработки *Code::Blocks*, которая легко устанавливается, удобна и проста в применении. Но вы можете использовать исходный код программ и в других средах. Например, в *Visual Studio* или *Borland C++ Builder*.

За исключением операций ввода-вывода в исходном коде нет специфических конструкций, поэтому с небольшими исправлениями его можно перенести на язык *с*.

Книга адресуется: школьникам изучающим *C++* на уроках или самостоятельно, учителям информатики и математики, любителям программирования и математики.

Валерий Рубанцев

.....

Условные обозначения, принятые в книге:

Дополнение или замечание

Требование или указание

Исходный код

Задание для самостоятельного решения

Заголовок проекта:

Проект

Исходные коды всех проектов находятся в папке `_CPPProjects`

Оглавление

От автора	3
Оглавление	6
Установка программы <i>Code::Blocks</i>	9
СТАРИННЫЕ ЗАДАЧИ	28
Египетские кошки	29
Шахматное число	36
Вавилонские ладони	39
Индийские пчёлы	41
Китайские кролики и фазаны	43
Вьетнамские буйволы	47
Индийские обезьяны	49
Индийские обезьяны 2	51
Жизнь Демохара	53
Греческие мешконосы	54
Индийское число	56
Пифагорейское число	58
Индийский храм	60
Египетские коровы	62
Римские адвокаты	64
Диофантово число	66
Арабские голуби	68
Персидские яблоки	70
Кахунский папирус	72
Берлинский папирус	73
Индийские квадраты	75
СТАРЫЕ ЗАДАЧИ	77
Чисто американская задача	78
Чисто французская задача	80
Репетитор	82
Американские цыпочки	84

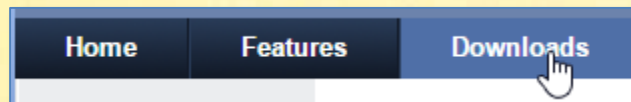
Американское наследство	86
Английский юмор	87
Русские яблоки	90
Турецкие долгожители	93
Чешские сливы	94
Французский покупатель	96
Болгарский парикмахер	98
Болгарские сливы	99
Немецкий вопрос	101
Индийские рупии	103
Русские гуси	104
Насос Эдисона	107
Китайская арифметика	109
Задача Этьена Безу	111
Кому сколько лет?	113
Ноги и головы	115
Задания для самостоятельного решения	117
СОВРЕМЕННЫЕ ЗАДАЧИ	120
Кузнечики	121
Бельгийские числа	124
Немецкая копилка	126
Ошибки	128
Коровы	130
Мешки с сокровищами	131
Повторяющиеся цифры	134
Квадраты 1,4,9	136
Кубики из кубиков	139
Бизнес на мышах	142
Гимнастический зал	145
Грузовые машины	147
Кувшин	149
Ещё один кувшин	151
Склады вание бумаги	153
Vogenschießen	158

Город.....	161
Наименьшее число	164
Трёхзначное число	166
Треугольные числа	167
Великолепная четвёрка.....	172
Рекурсивный тортик.....	175
Тортик.....	179
Столетие.....	182
Суперпростые числа	184
Числа Лишрел	189
Литература	194
Серия Программирование для детей	196
Серия Программирование на языке C# 5.0: Начальный уровень	201

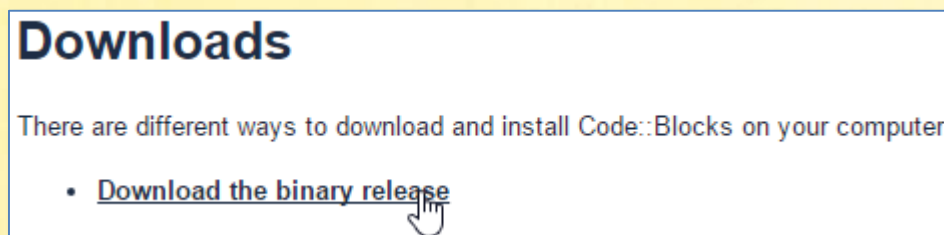
Установка программы *Code::Blocks*

Для программирования на *C++*, как, впрочем, и на других языках, нужна удобная среда разработки. Для этого часто используют программу *Visual Studio*, но её скачивание и установка занимает очень много времени. Поэтому я выбрал значительно менее громоздкую программу *Code::Blocks*.

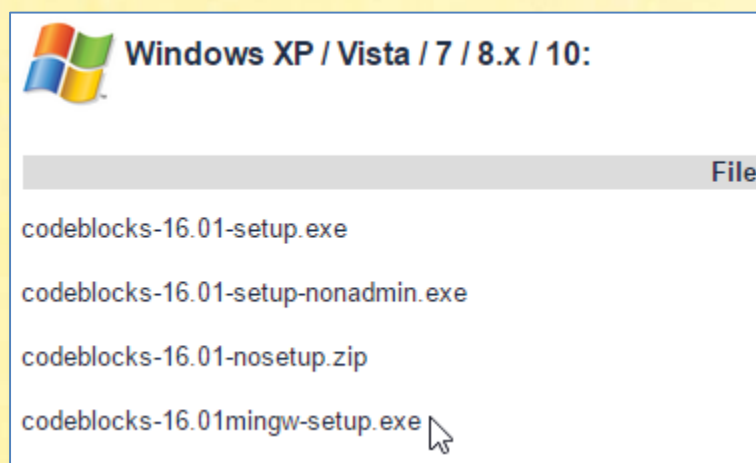
Откройте официальный сайт программы www.codeblocks.org. Нажмите кнопку **Downloads**:



На следующей странице щёлкните по строчке **Download the binary release**:



Нам нужен файл `codeblocks-16.01mingw-setup.exe`:



Выберите справа сайт для загрузки установочного файла: [Sourceforge.net or FossHub](#).

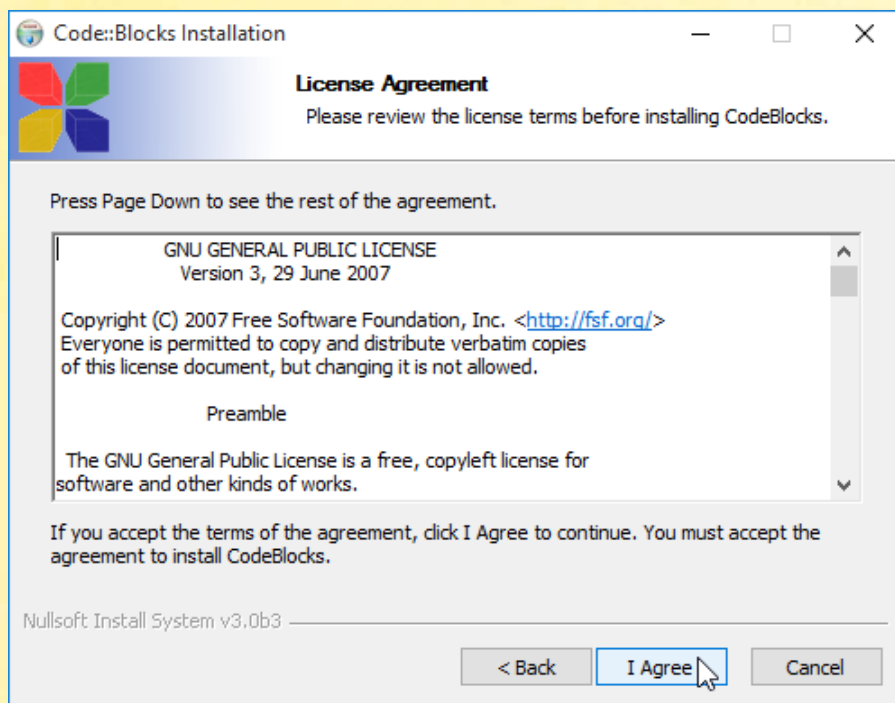
С сайта **FossHub** загрузка начинётся автоматически.

Установочный файл **codeblocks-16.01mingw-setup.exe** имеет размер около 80 МВ, и загружается очень быстро.

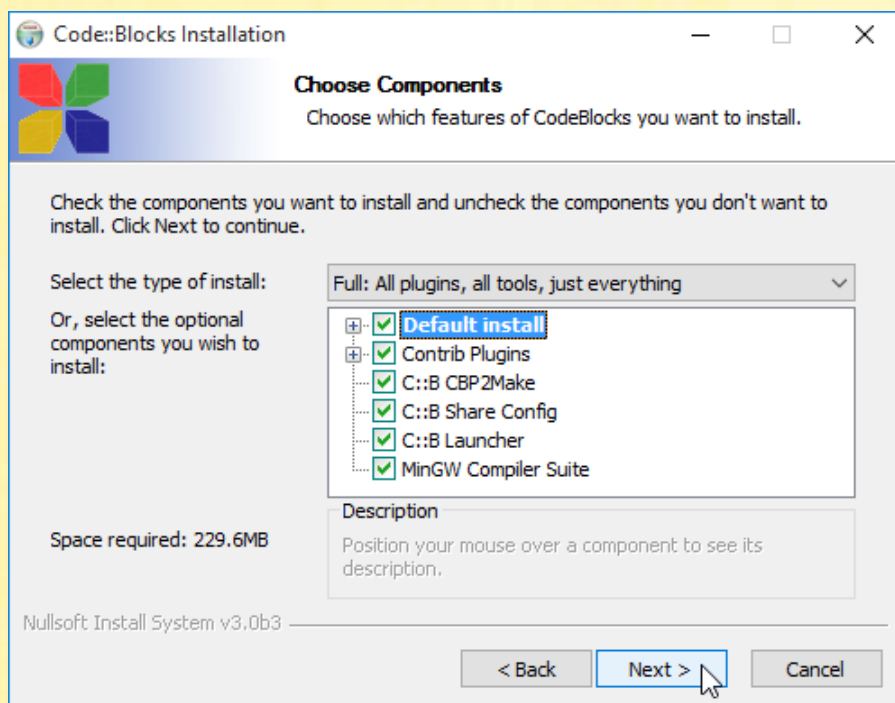
Запустите программу **codeblocks-16.01mingw-setup.exe**. В первом диалоговом окне просто нажмите кнопку **Next >**:



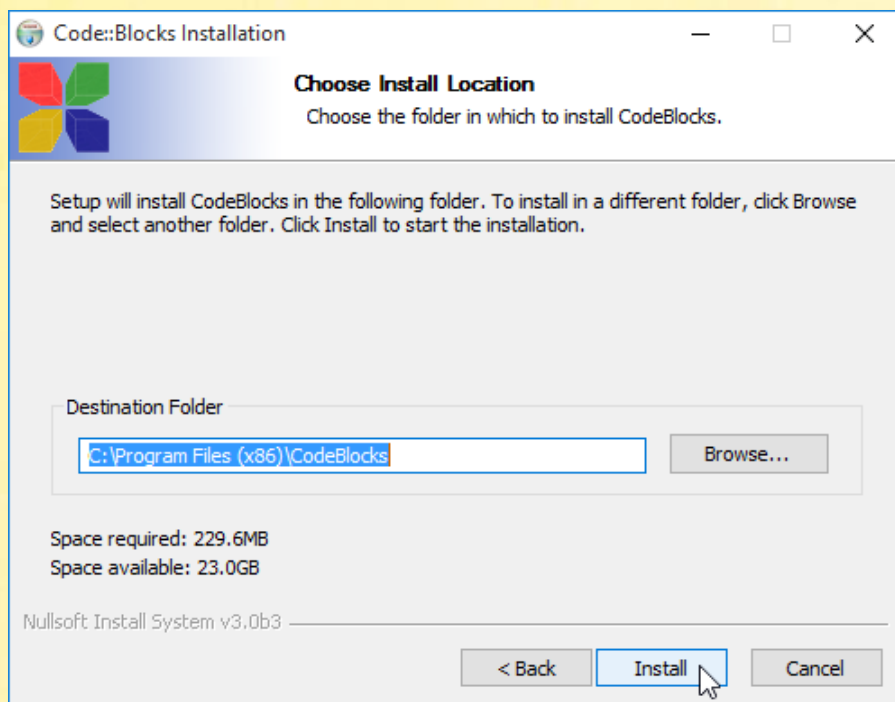
В следующем диалоговом окне вам предложат ознакомиться с лицензией на пользование программой. Это обычная формальная процедура, поэтому смело соглашайтесь со всеми условиями и нажимайте кнопку **I Agree**:



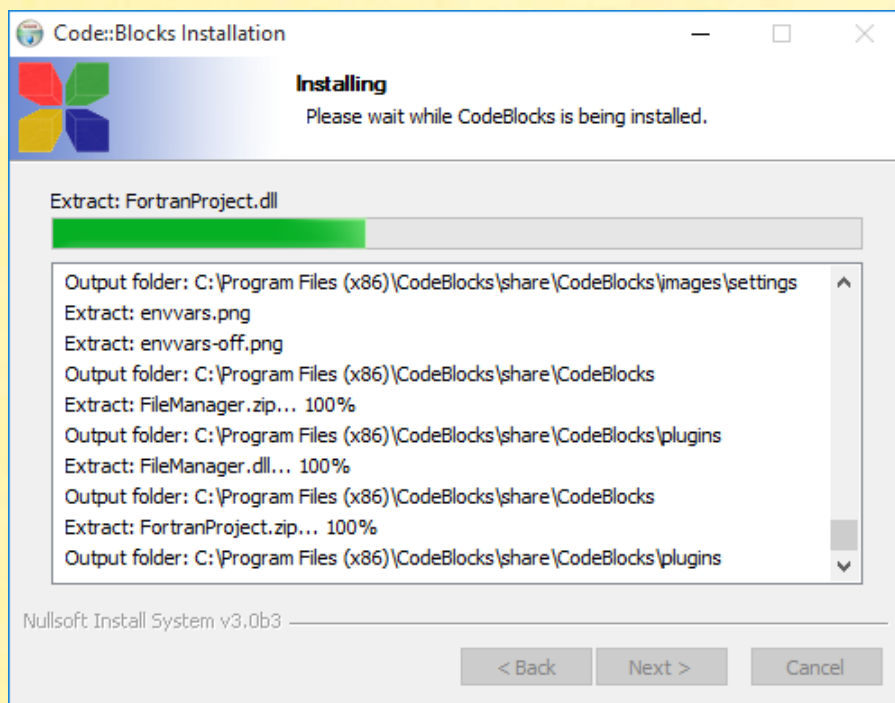
В третьем по счёту диалоговом окне нужно выбрать компоненты для установки на компьютер. Поскольку для всех компонентов требуется не очень много места на диске, а заранее трудно предвидеть, что именно вам понадобится, то нажмите кнопку **Next >**:



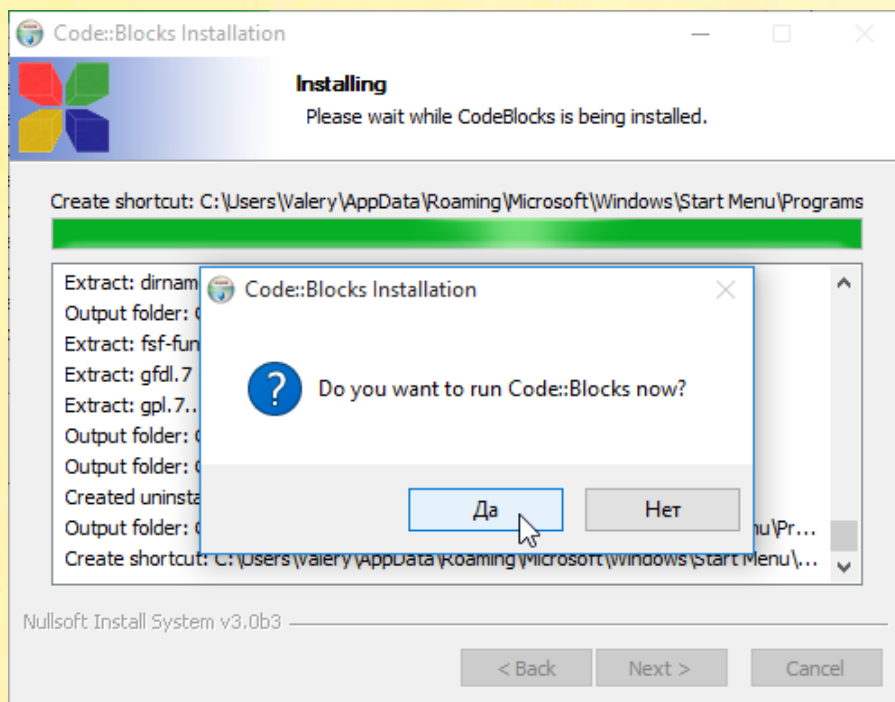
Программа сама знает, в какую папку на диске ей нужно устанавливаться, поэтому сразу нажимайте кнопку **Install**:



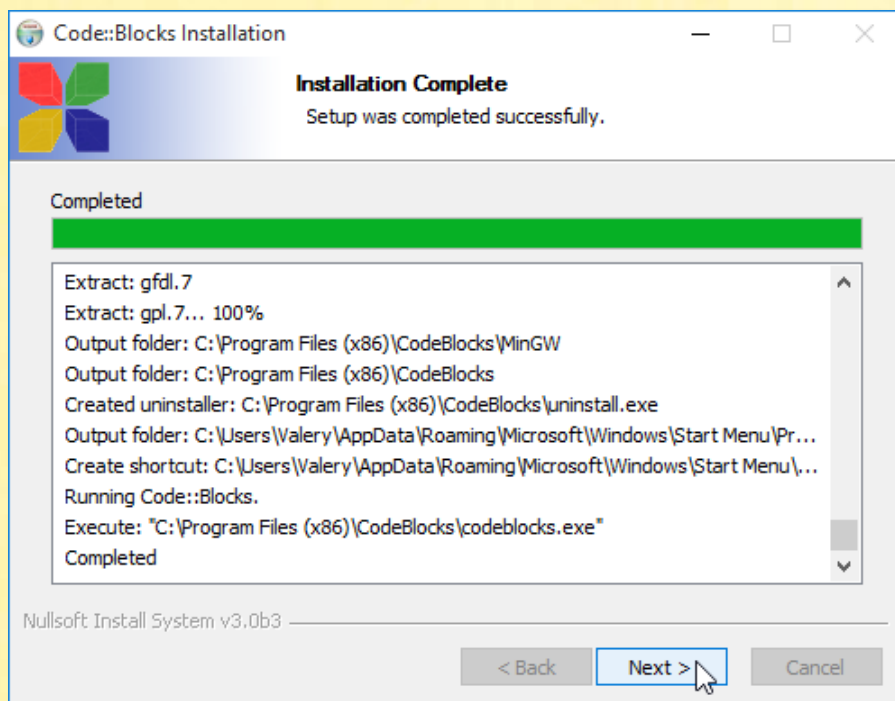
Ход установки показывает зелёная полоска:



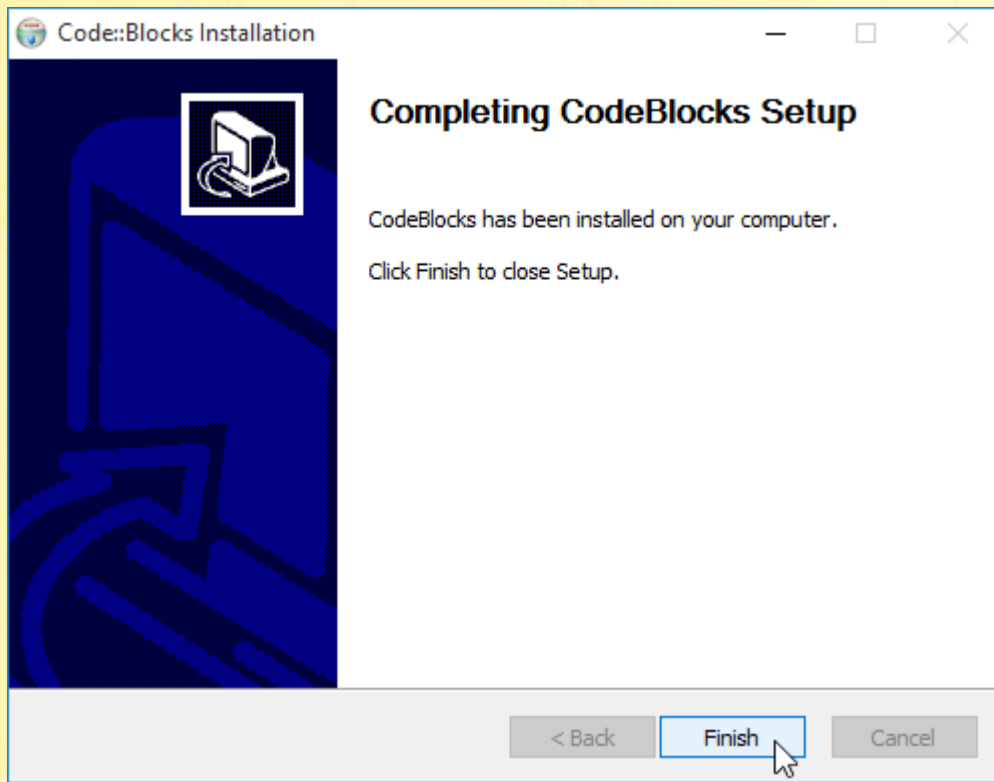
Через пару минут установка будет закончена, и вы сможете запустить среду разработки, нажав кнопку **Да**:



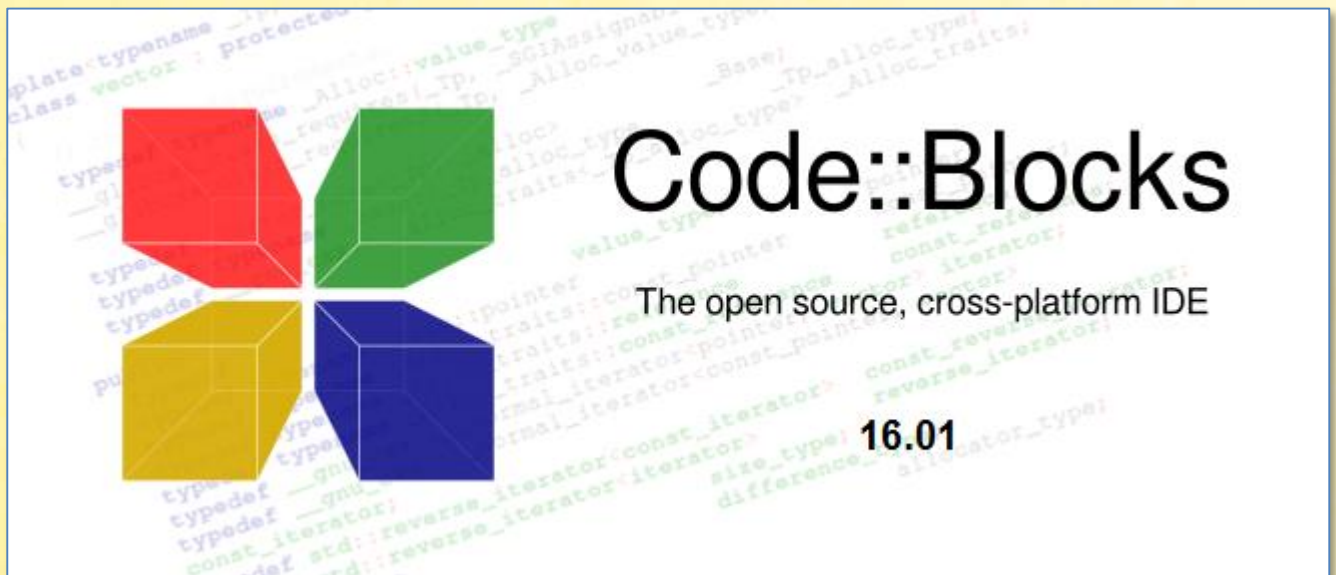
Пока программа запускается, нажмите кнопку **Next >** в диалоговом окне:



А затем кнопку **Finish**, чтобы закрыть все диалоговые окна:

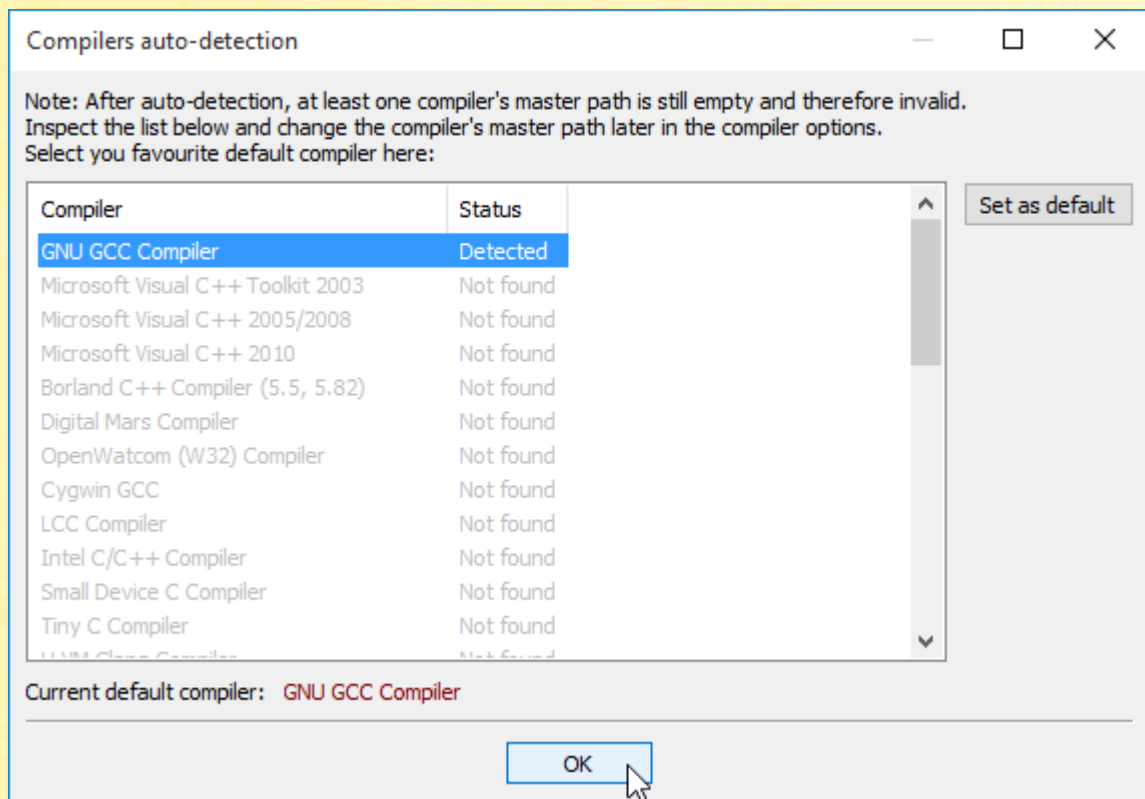


В это время на экране появится заставка программы **Code::Blocks**:



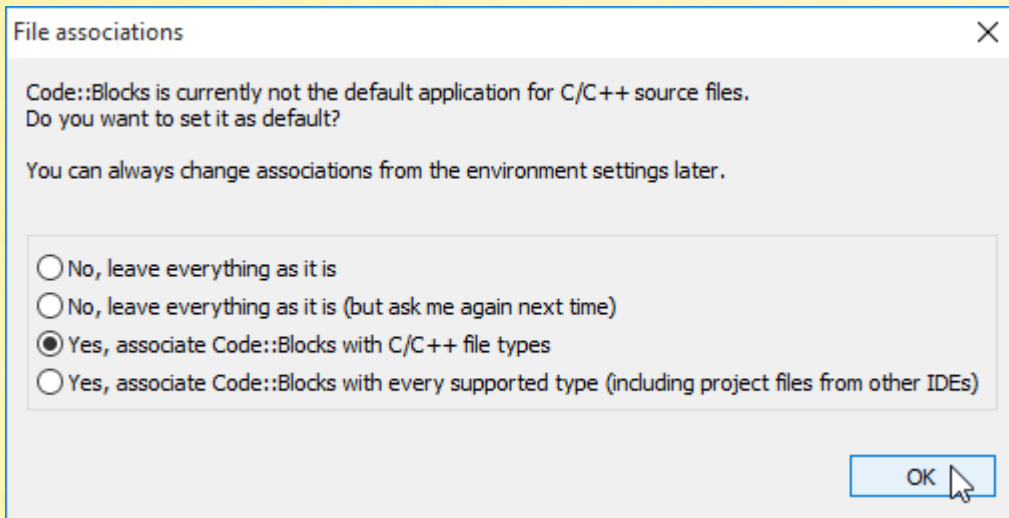
На ней можно прочитать, что это программа с открытым исходным кодом и предназначена для разработки программы на компьютерах с разными операционными системами. И то, и другое очень важно при выборе программ для работы.

Если на вашем компьютере не установлены другие компиляторы из списка, то будет использоваться **GNU GCC Compiler**, который нам и нужен:

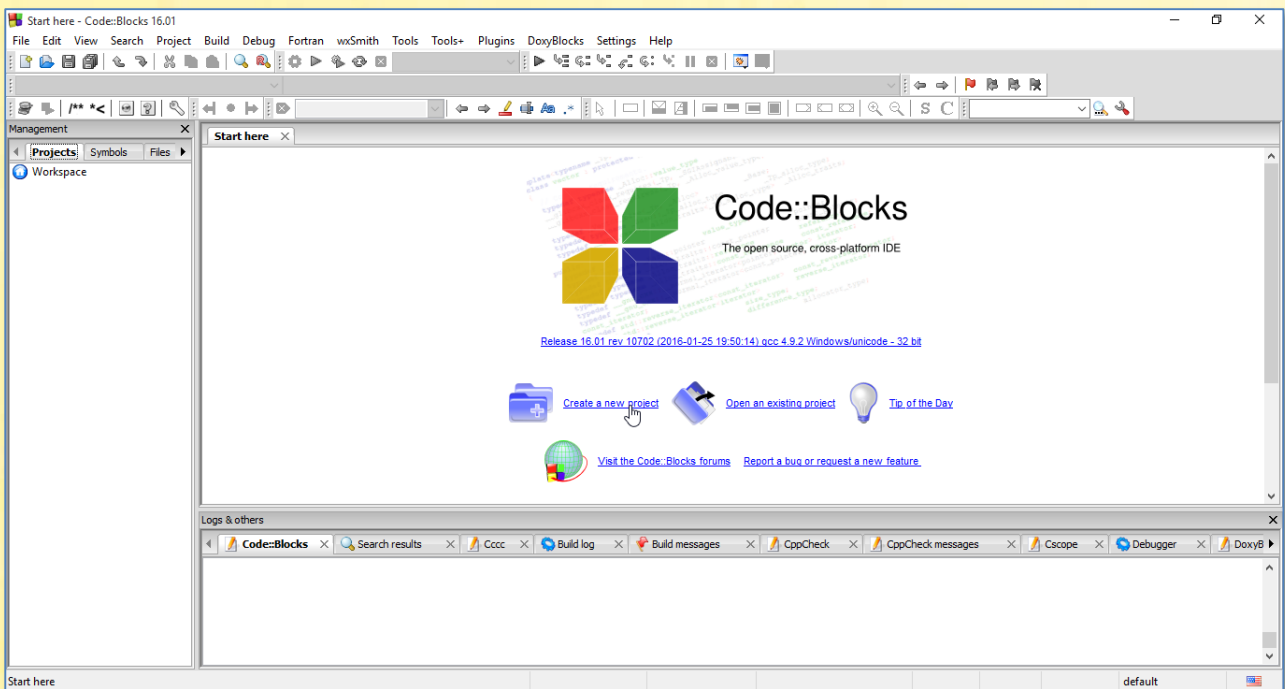


В противном случае выберите другой компилятор и нажмите кнопку **Set as default**, чтобы он использовался средой разработки при запуске ваших программ.

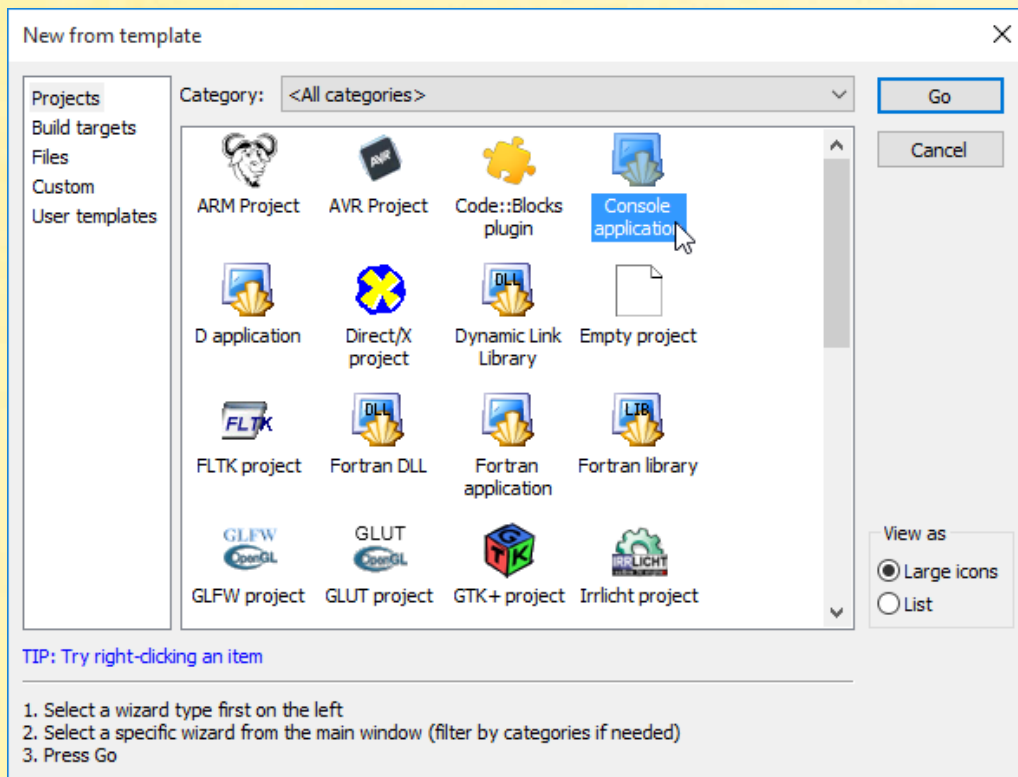
Если у вас на компьютере нет других сред разработки программ на *C++*, то выберите третий пункт в диалоговом окне *File associations* и нажмите кнопку **OK**:



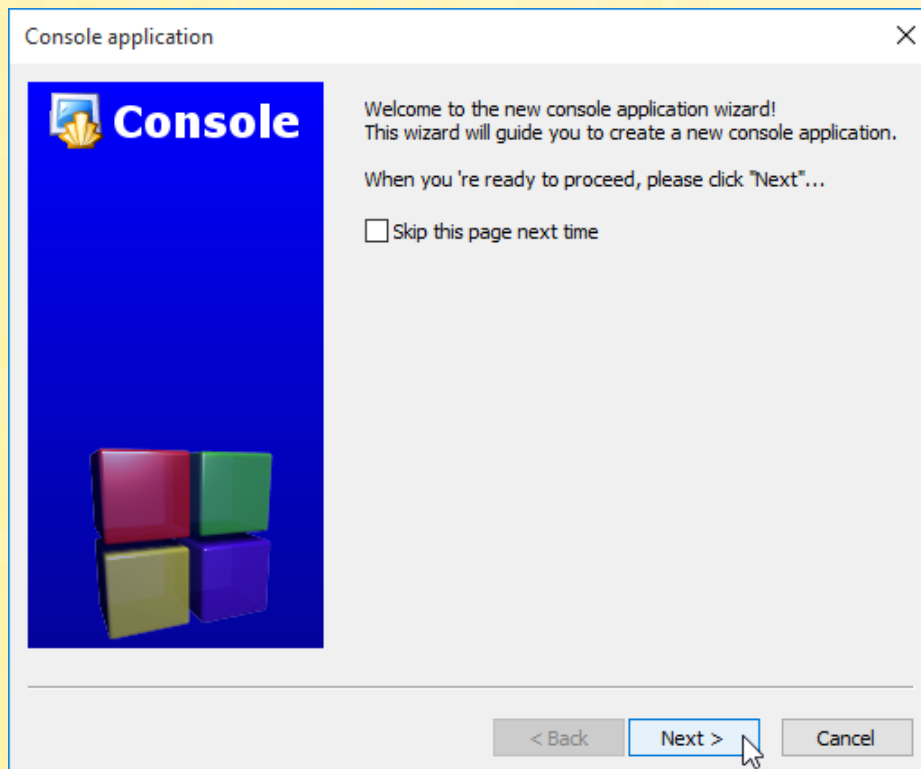
Так как никаких проектов у нас пока нет, то щёлкните по строчке **Create a new project**:



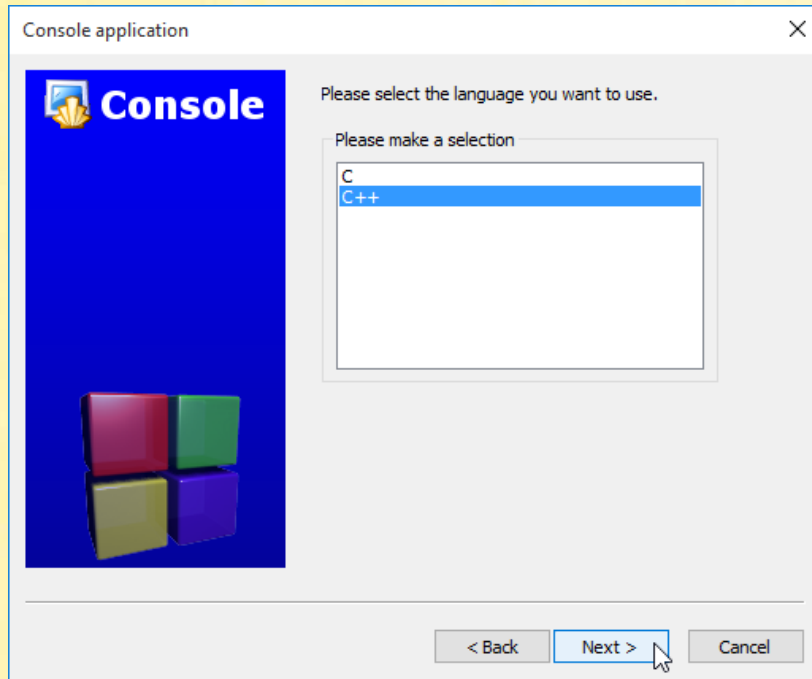
Для решения задач вполне достаточно **консольных приложений**, поэтому выберите шаблон *Console application* и нажмите кнопку **Go**:



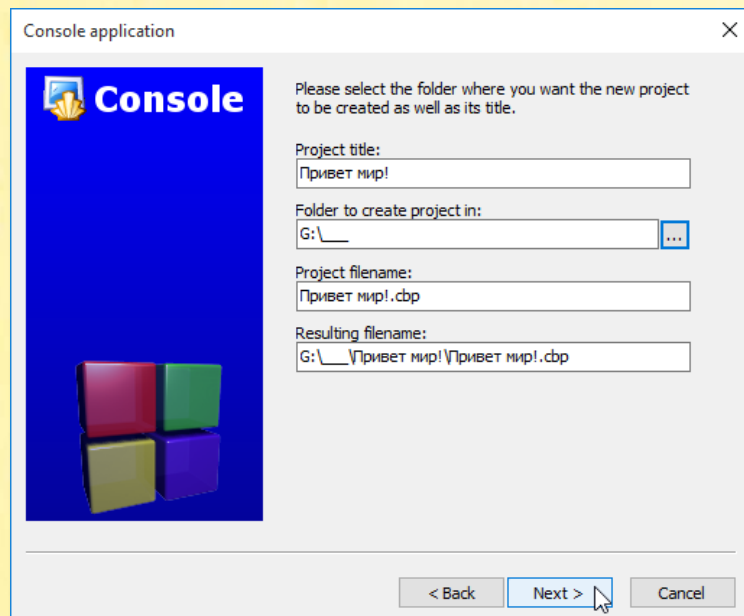
В следующем диалоговом окне нажмите кнопку **Next >**:



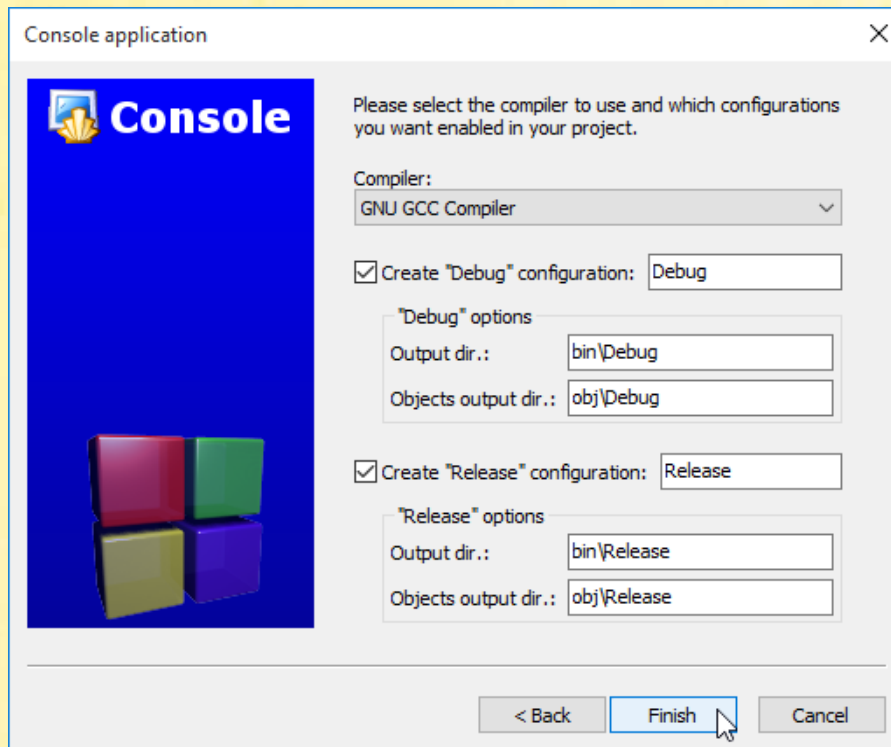
Мы будем писать на языке *C++*, который выбран по умолчанию, но вы можете перейти на более старый язык *сi*:



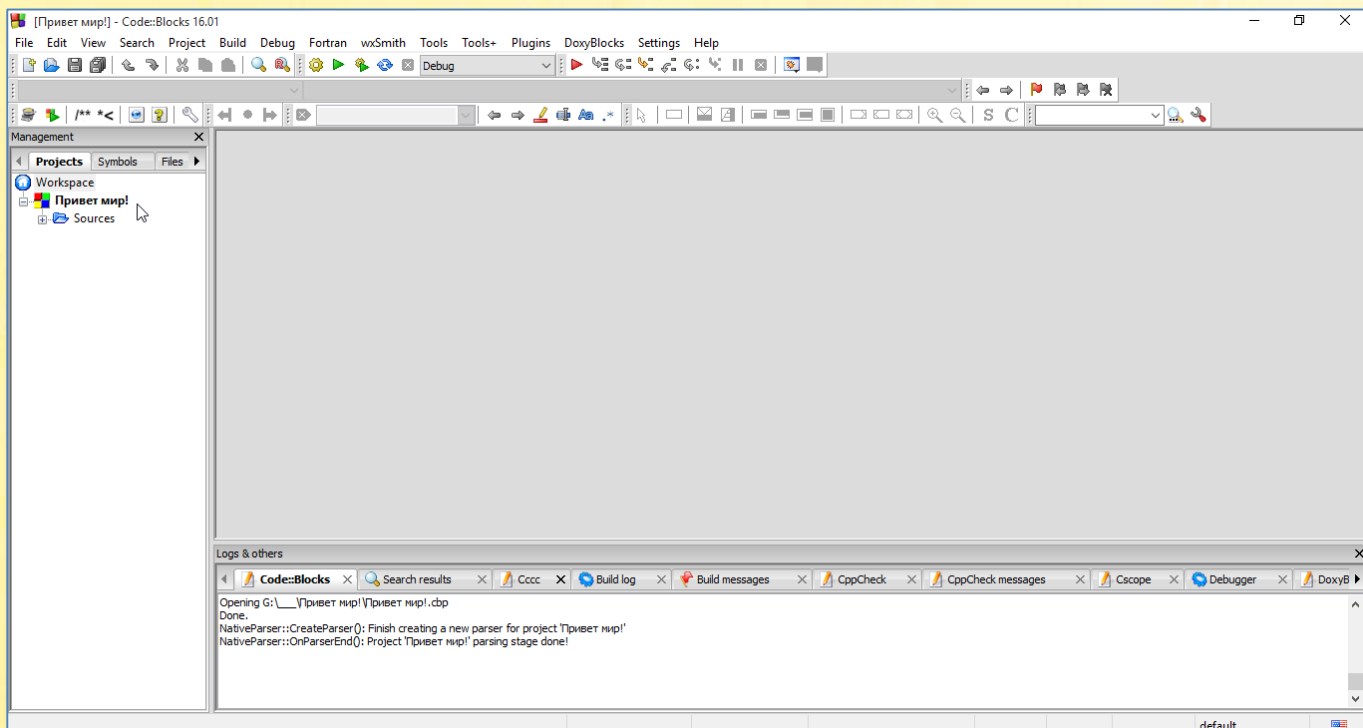
Затем наберите в текстовом поле **Project title** название проекта. По традиции в первом проекте приветствуют Мир, поэтому я и выбрал такое имя для него. Нажмите кнопку с тремя точками и выберите готовую или создайте новую папку для хранения ваших проектов и нажмите кнопку **Next >**:



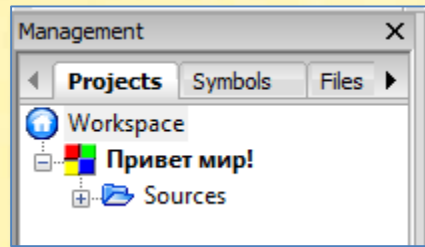
И в последнем диалоговом окне просто нажмите кнопку **Finish**:



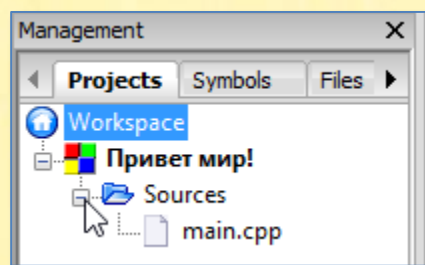
На этом подготовка закончена, и открывается **Главное окно** среды разработки:



На панели **Management** вы можете видеть наш проект:

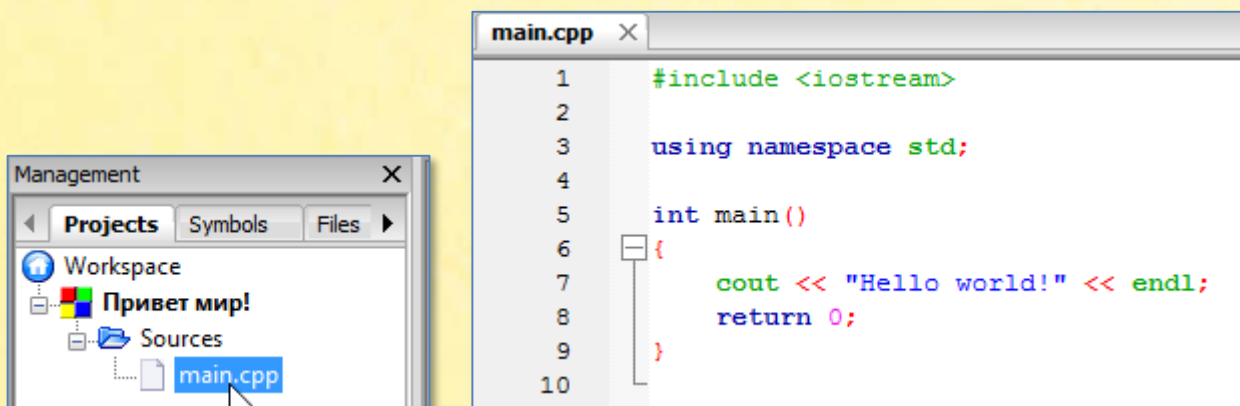


Откройте папку **Sources**, щёлкнув мышкой по квадрату с плюсом:



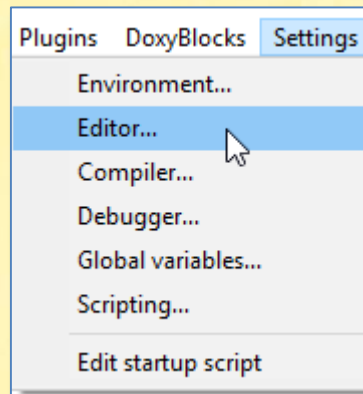
В ней только один файл – **main.cpp**. Это **главный файл** любого проекта.

Щёлкните по названию файла, чтобы открыть его в *Редакторе кода*:

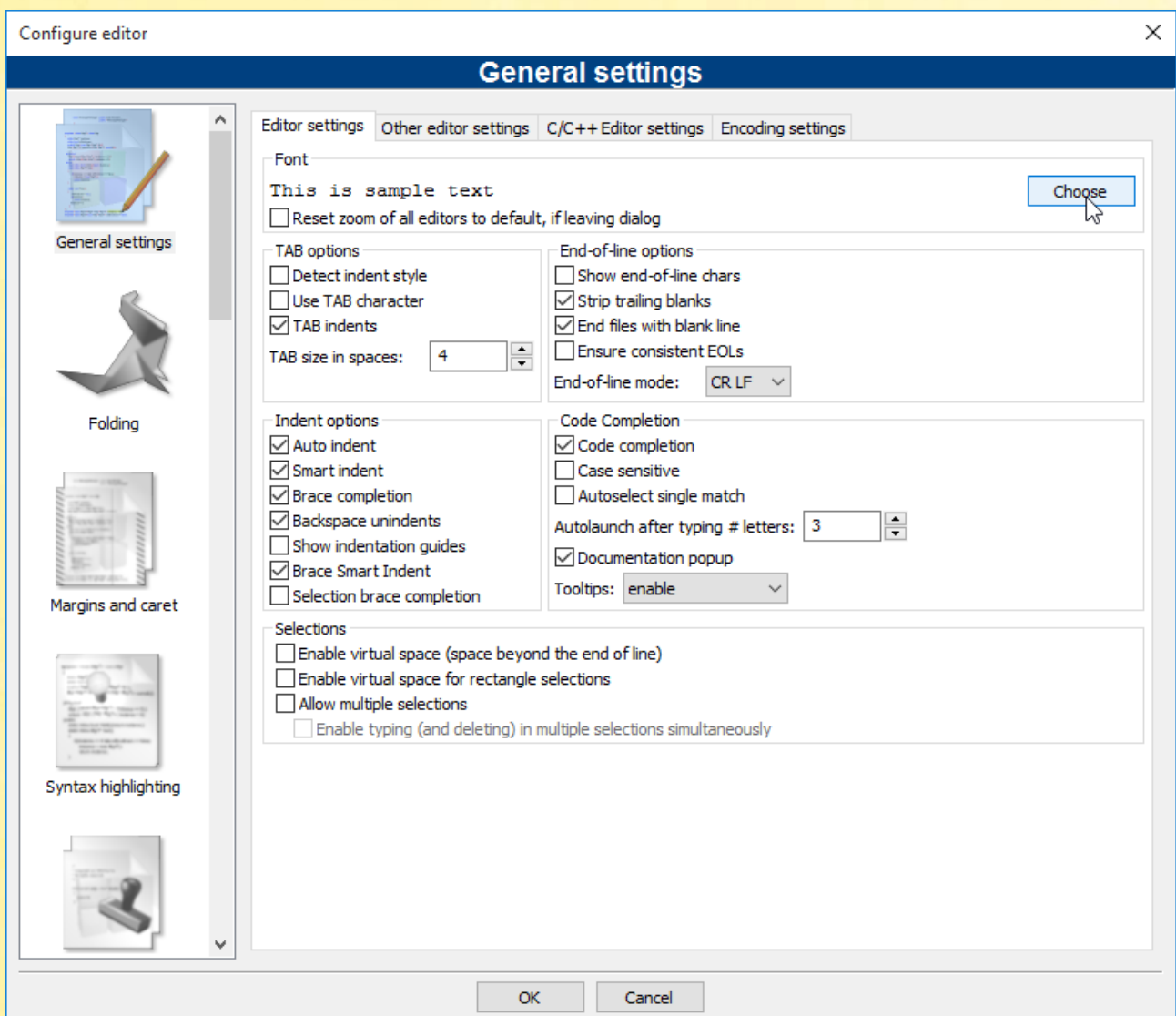


Сразу бросаются в глаза мелкие буквы, которые очень трудно читать.

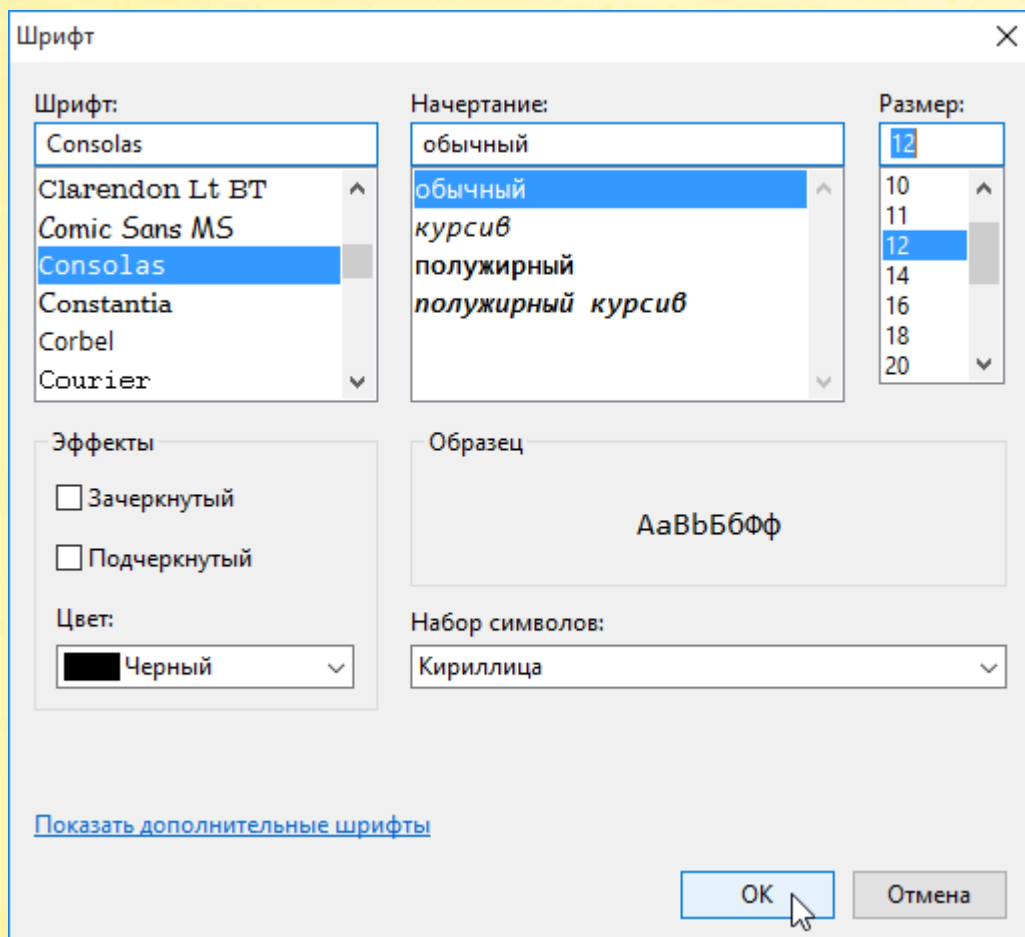
Выполните команду меню **Setting** → **Editor...**:



В диалоговом окне *Configure editor* нажмите кнопку **Choose**:



Выберите шрифт **Consolas** размером **12** пунктов и нажмите кнопку **OK**.

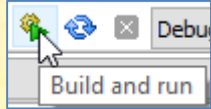


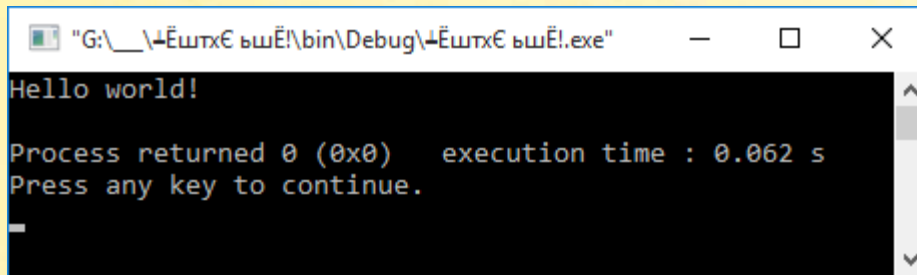
Теперь исходный текст виден как на ладони:

```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
10
```

Я только показал, как можно изменить шрифт в редакторе. Вы, конечно, должны выбрать шрифт по своему вкусу.

Как вы видите, шаблон консольного приложения уже готов напечатать привет-

ствие миру на английском языке. Нажмите кнопку  на *Панели инструментов*. Откроется консольное окно с заданной строкой:



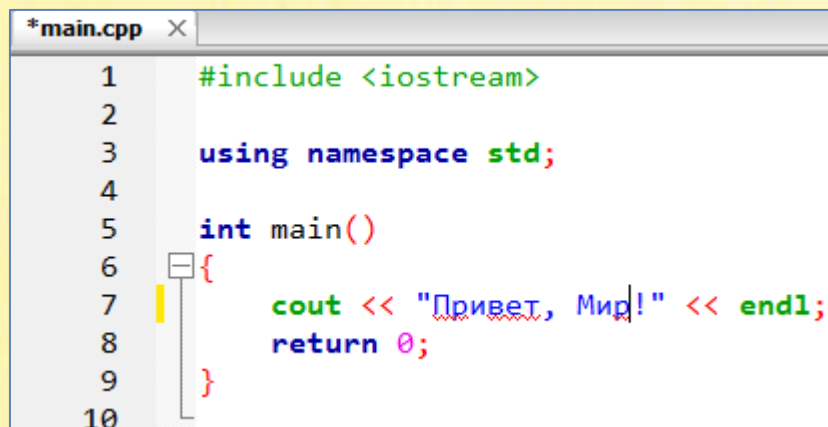
```
"G:\_ \-Ёштх€ ЫшЁ!\bin\Debug\_-Ёштх€ ЫшЁ!.exe"
Hello world!

Process returned 0 (0x0)    execution time : 0.062 s
Press any key to continue.
-
```

В **заголовке окна** показан полный путь к исполняемому файлу на диске. Русские буквы напечатаны неверно, из чего следует, что названия проектов желательно писать латинскими буквами.

Закройте консольное окно.

Но мы хотим приветствовать Мир на родном языке, поэтому исправляем строчку:



```
*main.cpp x
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       cout << "Привет, Мир!" << endl;
8       return 0;
9   }
10
```

Опять нажимаем кнопку **Build and run** и получаем тарабарскую грамоту вместо приветствия:

A screenshot of a Windows command prompt window. The title bar shows the file path: "G:_...\bin\Debug\...". The main text in the window is: "Привет, Мир!" followed by "Process returned 0 (0x0) execution time : 0.078 s" and "Press any key to continue.".

Если вы хотите выводить в консольное окно надписи на русском языке, то добавьте вызов функции `SetConsoleOutputCP` с номером русской кодировки и директиву `#include<windows.h>`:

A screenshot of a Visual Studio code editor window titled "main.cpp". The code is as follows:

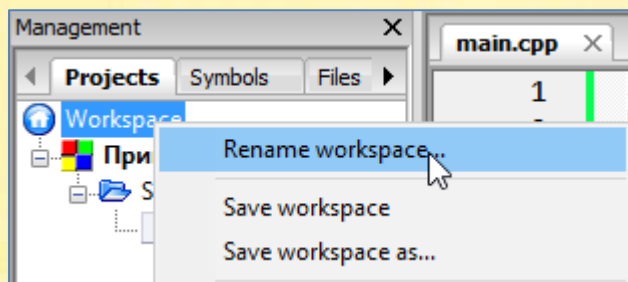
```
1 #include <iostream>
2 #include<windows.h>
3
4 using namespace std;
5
6 int main()
7 {
8     SetConsoleOutputCP(1251);
9     cout << "Привет, Мир!" << endl;
10    return 0;
11 }
12
```

Теперь надпись выводится правильно:

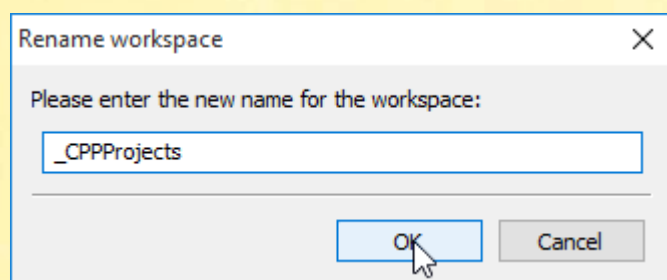
A screenshot of a Windows command prompt window. The title bar shows the file path: "G:_...\bin\Debug\...". The main text in the window is: "Привет, Мир!" followed by "Process returned 0 (0x0) execution time : 0.047 s" and "Press any key to continue.".

Все проекты лучше хранить в одной папке на диске и открывать их в одной **группе**. Сейчас она называется **Workspace** (*рабочее пространство, рабочая среда*), но мы её переименуем, чтобы назначение было более понятно.

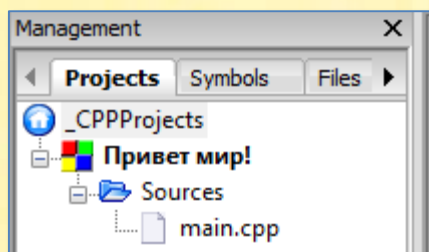
Вызовите контекстное меню на названии группы проектов и щёлкните по строчке **Rename workspace...**:



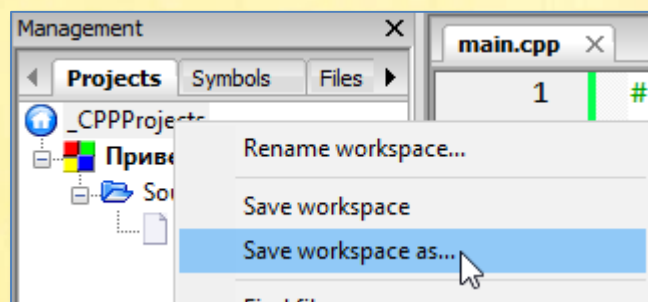
Переименуйте группу и нажмите кнопку **OK**:



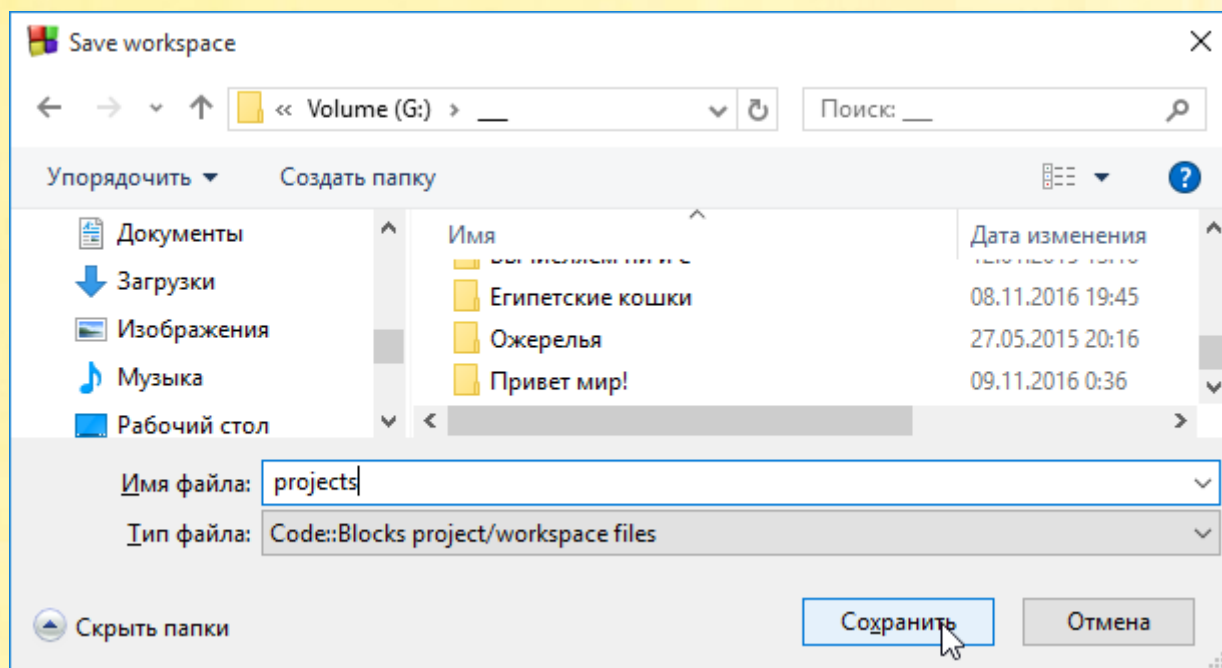
Получилось замечательно:



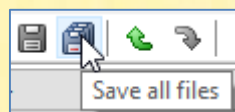
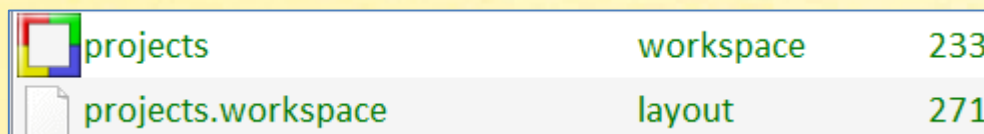
Но свойства группы нужно сохранить на диске, в папке с проектами. Опять вызовите контекстное меню группы и нажмите строчку **Save workspace as...**:



В открывшемся диалоговом окне задайте имя для файла группы и нажмите кнопку Сохранить:

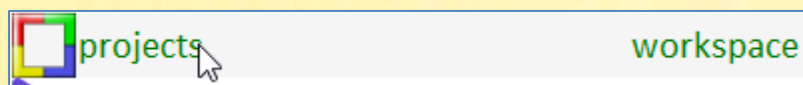


На диске появятся 2 новых файла:

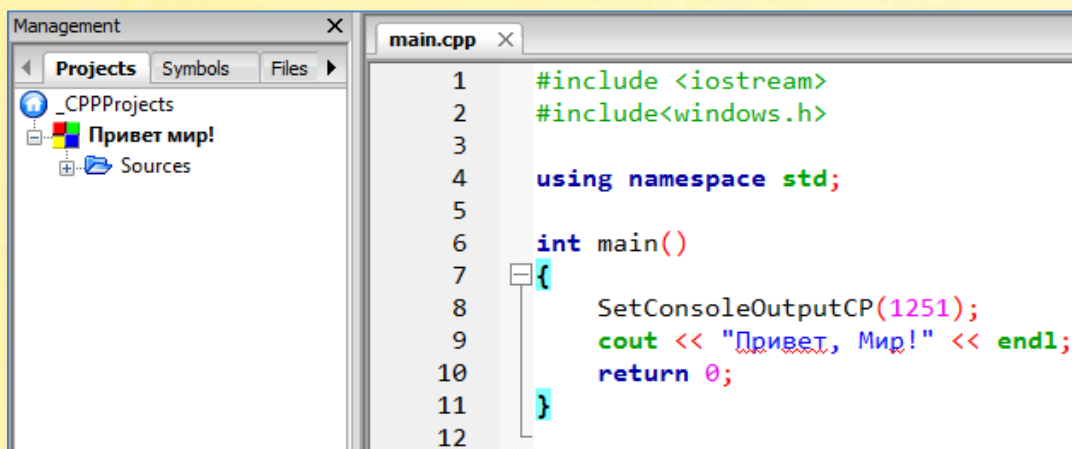


Нажмите кнопку , чтобы сохранить изменения в файле проекта *Привет мир!*

Закройте среду разработки. Перейдите в папку с проектами и щёлкните по файлу `projects.workspace`:



Вся группа проектов (сейчас в ней только один проект) откроется в среде разработки:



Последний штрих в первой программе.

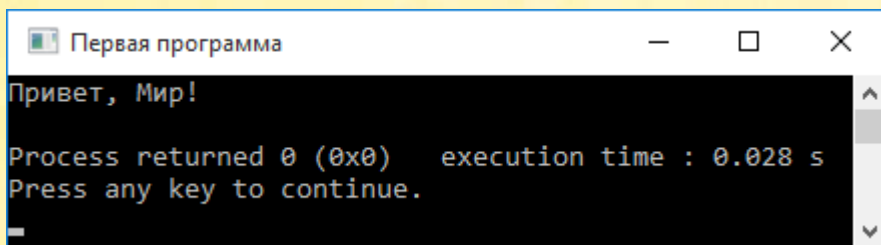
Чтобы напечатать название консольного окна русскими буквами, используйте функцию **system**. В скобках запишите в кавычках слово *title* и название заголовка окна. Теперь наша программа должна выглядеть так:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Первая программа");
    cout << "Привет, Мир!" << endl;
    return 0;
}
```

Мы научили говорить *C++* по-русски:



На этом первый проект можно считать законченным!

СТАРИННЫЕ ЗАДАЧИ

VON EULER BIS ZUR GEGENWART

История математики насчитывает более 6 тысяч лет! Конечно, занимательные задачи появились не вдруг и не сразу, ведь нашим далёким предкам ещё нужно было научиться считать на пальцах и придумать цифры для записи чисел.

Но самым древним «рукописям» - глиняным табличкам и папирусным свиткам - больше 4,5 тысяч лет, а мы находим на них не только «учебные» арифметические задачи, в которых требуется преобразовать какое-либо выражение или вычислить его значение, но и вполне занимательные задачи, не имеющие прямого практического значения.

Главное отличие занимательных задач от учебных заключается в том, что никто не заставляет их решать. Они настолько интересны сами по себе, что их непременно хочется решить.

 Springer

В этой главе собраны древние и старинные занимательные задачи, и я надеюсь, что они заинтересуют вас точно так же, как и любителей математики много столетий и тысячелетий тому назад.

Египетские кошки

Задача из книги Математический фольклор (Д1):

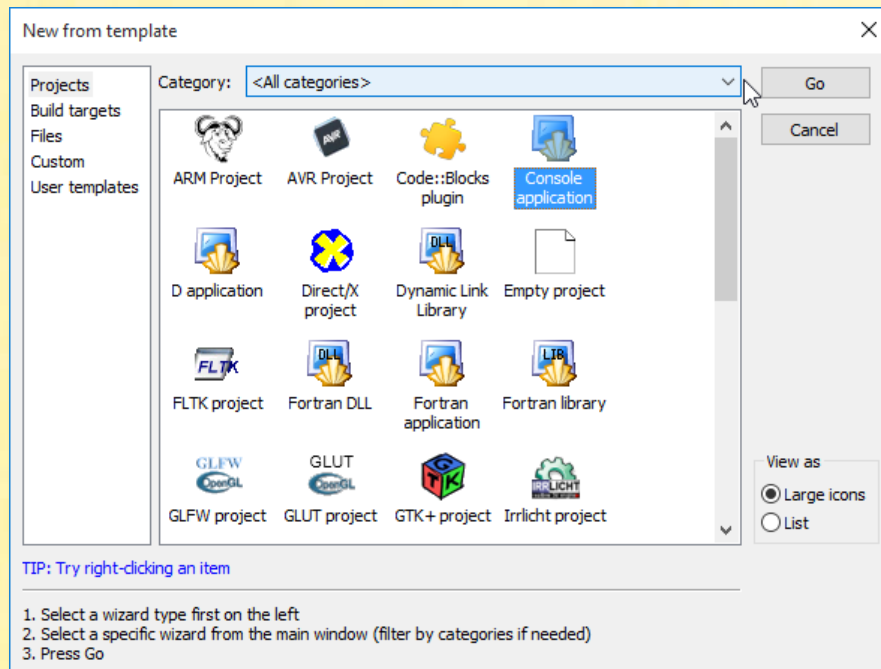
Каждый из 7 человек имеет 7 кошек. Каждая кошка съедает по 7 мышек. Каждая мышка за одно лето может уничтожить 7 ячменных колосков, а из зёрен одного колоска может вырасти 7 горстей ячменного зерна.

Сколько горстей зерна ежегодно спасается благодаря кошкам?

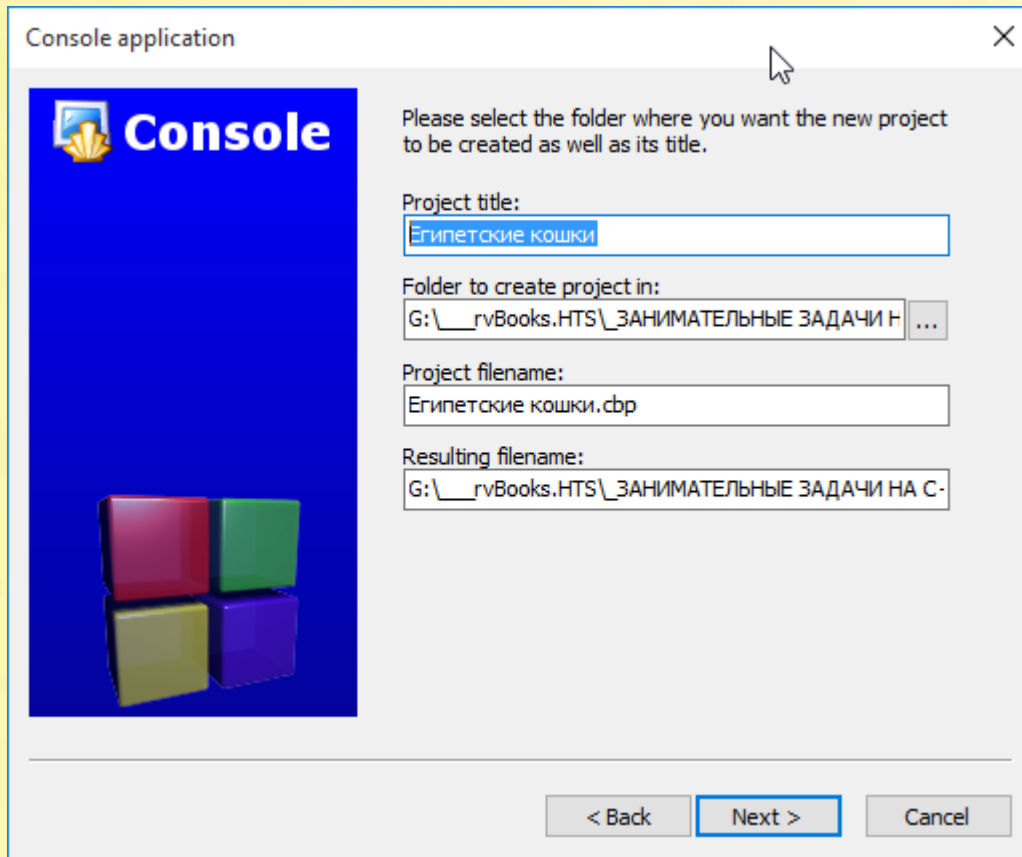


Эта популярная в Древнем Египте задача сейчас нам кажется очень простой, ведь чтобы решить её, достаточно найти произведение пяти семёрок.

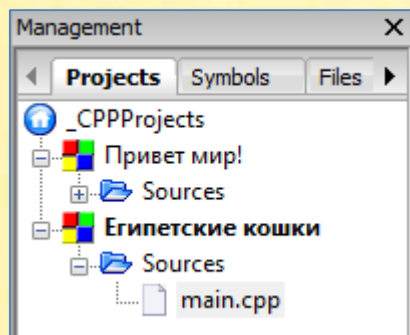
Откройте группу проектов щелчком по файлу *projects.workspace* в папке с проектами. Выполните команду меню **File** → **New** → **Project...** В диалоговом окне **New from template** выберите **Console application** и нажмите кнопку **Go**:



Дальнейшие шаги вам уже известны, поэтому остановимся только на диалоговом окне **Console application**, в котором нужно ввести *имя нового проекта*:



В группе проектов появилась папка **Египетские кошки**. Она выделена **жирным** шрифтом, который означает, что этот проект **активный**, то есть он будет компилироваться по команде *Build and run*:



Откройте главный файл **main.cpp** в *Редакторе кода*, дважды щёлкнув по нему. Это шаблон для всех консольных приложений.

```
main.cpp x *main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
```

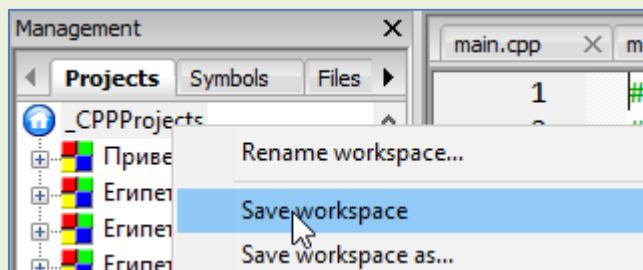
Обратите внимание, что в Редакторе кода сейчас открыты 2 главных файла наших проектов. Звёздочка перед названием файла сигнализирует, что изменения не сохранены на диске. Не забывайте регулярно (особенно перед запуском программы!) сохранять файлы на диске, нажимая кнопки **Save** и **Save all files**:



Первая кнопка сохраняет только открытый в Редакторе кода файл, а вторая – все изменённые файлы во всех проектах.

Так как Редактор кода работает нестабильно, то после добавления нового проекта сразу же сохраняйте изменения во всей группе.

Для этого вызовите контекстное меню на названии группы и выполните команду **Save workspace**:



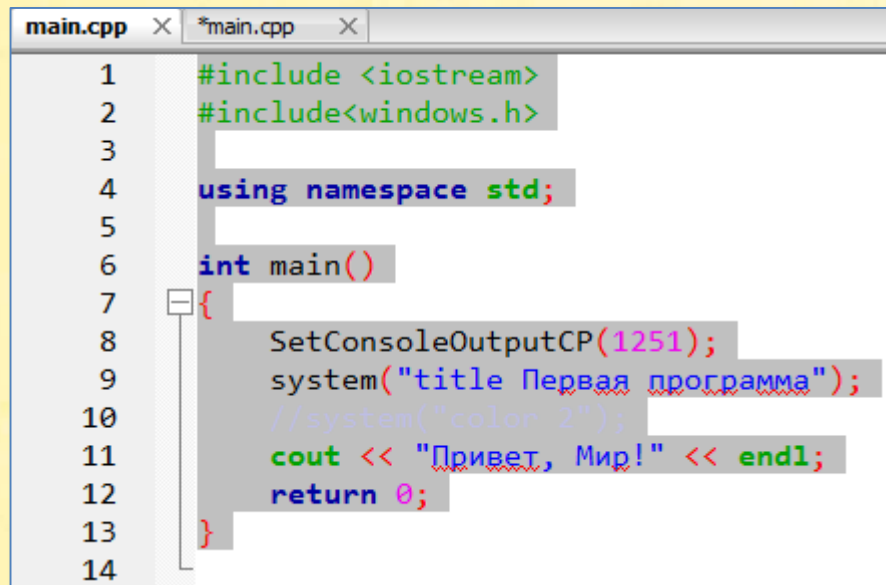
Можно переключаться между файлами, нажимая закладки с их названиями.

Откройте главный файл первой программы. Нажмите клавиши **Ctrl + A**, чтобы **выделить** весь текст:

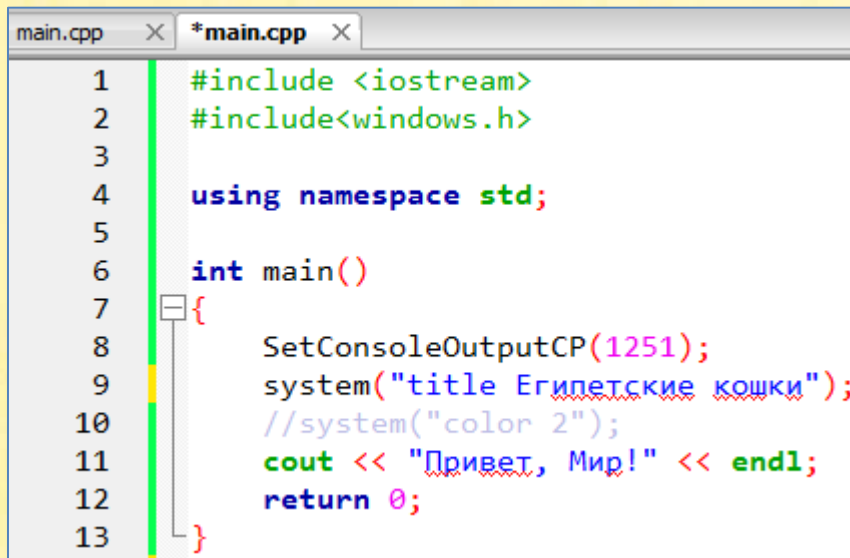
Скопируйте его в буфер обмена клавишами **Ctrl + C**. Вернитесь в главный файл новой программы. Выделите весь код и нажмите клавиши **Ctrl**

+ V, чтобы **вставить** код из буфера. Эти команды используются во всех текстовых редакторах, поэтому вам должны быть хорошо известны.

Программисты не любят писать один и тот же код дважды, поэтому просто копируют готовые куски кода в новую программу, что мы и сделали. Внесём изменения в программе и исправим заголовок консольного окна:



```
main.cpp x *main.cpp x
1  #include <iostream>
2  #include<windows.h>
3
4  using namespace std;
5
6  int main()
7  {
8      SetConsoleOutputCP(1251);
9      system("title Первая программа");
10     //system("color 2");
11     cout << "Привет, Мир!" << endl;
12     return 0;
13 }
```



```
main.cpp x *main.cpp x
1  #include <iostream>
2  #include<windows.h>
3
4  using namespace std;
5
6  int main()
7  {
8      SetConsoleOutputCP(1251);
9      system("title Египетские кошки");
10     //system("color 2");
11     cout << "Привет, Мир!" << endl;
12     return 0;
13 }
```

Заголовок напечатан правильно:

A screenshot of a Windows console window titled "Египетские кошки". The window has a black background and white text. The text displayed is: "Привет, Мир!" followed by "Process returned 0 (0x0) execution time : 0.051 s" and "Press any key to continue." The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Можно решать задачу!

Проще всего решить задачу, найдя произведение пяти семёрок:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Египетские кошки");
    int res = 7 * 7 * 7 * 7 * 7;
    cout << "Число горстей зерна равно: " << res << endl;
    return 0;
}
```

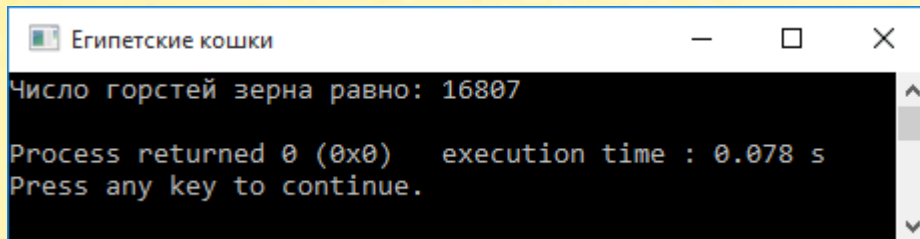
Запускаем программу и получаем **ответ**:

A screenshot of a Windows console window titled "Египетские кошки". The window has a black background and white text. The text displayed is: "Число горстей зерна равно: 16807". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

В переводе с древнеегипетских горстей на современные килограммы это составляет примерно **1350**.

Для древних египтян эта задача была, несомненно, более трудной, чем для нас. Но зато благодаря ей мы узнали, за что древние египтяне так любили кошек!

В папке **Египетские кошки** → **bin** → **Debug** вы найдёте *исполняемый файл* программы **Египетские кошки.exe**. Щёлкните по нему, чтобы запустить программу. Консольное окно только появится на экране и тут же исчезнет. Всё дело в том, что при запуске программы из среды разработки окно закрывается только после нажатия на любую клавишу, о чём сообщает надпись *Press any key to continue*:



```
Египетские кошки
Число горстей зерна равно: 16807
Process returned 0 (0x0) execution time : 0.078 s
Press any key to continue.
```

В скомпилированной программе окно закрывается сразу после того как весь код выполнен. Чтобы **окно оставалось на экране** до нажатия клавиши ВВОД (ENTER), добавьте перед строкой `return 0;` оператор:

`cin.get();`

Конечно, найти произведение пяти семёрок несложно. А что делать, если умножений было бы больше. Тогда лучше воспользоваться циклом *for*:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Египетские кошки");
    int res = 1;
    for (int i = 1; i <= 5; ++i)
        res *= 7;
    cout << "Число горстей зерна равно: " << res << endl;

    return 0;
}
```

Так можно найти произведение любого числа семёрок. Но тут надо помнить, что тип *int* не может хранить большие числа.

В современной математике произведение пяти семёрок записывают сокращённо: 7^5 . Читать это выражение следует так: *семь в пятой степени*. Семёрка – это **основание** степени, пятёрка – **показатель**, а всё выражение в целом - **степень**. Для вычисления степеней нужно присоединить к проекту *математическую библиотеку*:

```
#include <math.h>
```

В этой библиотеке имеется функция **pow**, которой нужно передать основание и показатель степени:

```
pow (7, 5)
```

Эта функция возвращает результат типа *double*. Но присваивая его переменной типа **int** мы получим целое значение:

```
#include <iostream>
#include<windows.h>
#include <math.h>

using namespace std;

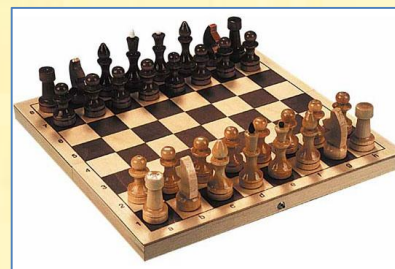
int main()
{
    SetConsoleOutputCP(1251);
    system("title Египетские кошки");

    int res = pow (7, 5);
    cout << "Число горстей зерна равно: " << res << endl;

    return 0;
}
```

Шахматное число

Вам, вероятно, известна легенда про изобретателя шахмат, которого правитель Индии, царь Шерам решил отблагодарить за интересную игру и предложил ему самому назначить цену.



И легенда, и задача встречаются во многих книгах по занимательной математике. Например, вы можете прочитать о ней в книге В. Литцмана *Великаны и карлики в мире чисел*, на странице 26.



Изобретатель - его звали Сета - попросил заплатить ему зерном. А зёрнышки посчитать так: на первую клетку шахматной доски положить 1 зерно пшеницы, на вторую 2, на третью 4, на четвёртую 8, и так далее, каждый раз увеличивая число зёрен в 2 раза.

На первый взгляд кажется, что в итоге получится всего горстка пшеницы. Так и подумал царь Шерам и приказал выдать скромному изобретателю гораздо больше – целый мешок пшеницы. Но тот попросил отмерить ровно столько зёрен, сколько получится в результате вычислений.

Шерам не стал спорить с изобретателем и повелел точно вычислить, сколько тому причитается зёрен. Когда придворные математики закончили свои долгие вычисления, то оказалось, что зёрен должно быть так много, что их просто не могло быть ни у Шерама, ни у кого другого на земле.

Если отбросить «мелкие подробности», то нам нужно найти сумму чисел:

$$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \dots + 2^{63}$$

Эти числа образуют **геометрическую прогрессию**, сумма которой равна $2^{64} - 1$.

При помощи операции возведения в степень мы без труда решим эту знаменитую шахматную задачу. На бумаге это трудное и долгое занятие, а вот написать программу на *C++* можно за несколько минут:

```
#include <iostream>
#include<windows.h>
#include<math.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Шахматное число");

    unsigned long long num = pow(2, 64) - 1;

    cout << " Шахматное число = " << num << endl;
    cout << endl;

    return 0;
}
```

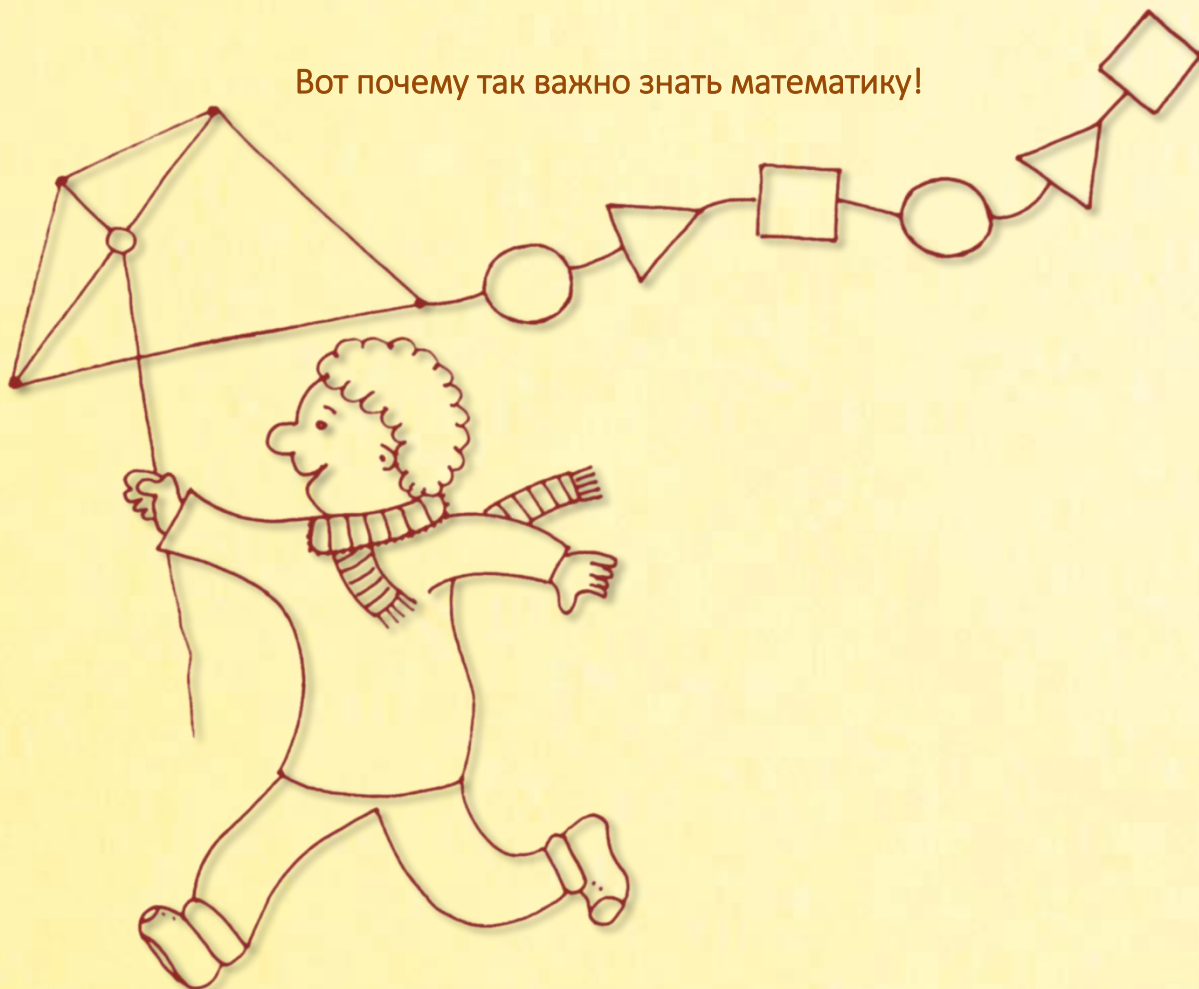
Степень мы легко вычислим с помощью функции `pow` математического класса `C++`, но нужно учесть, что для хранения такого длинного числа потребуется переменная беззнакового типа `unsigned long long`.

А число такое - 18 446 744 073 709 551 615:

```
Шахматное число  
Шахматное число = 18446744073709551615
```

Но число хоть и большое, но непонятное. Давайте переведём зёрна в тонны. Если принять, что зёрнышко весит 0,065 грамма, то общий вес составит около 1,2 триллиона тонн. Это больше, чем собрало человечество за всю свою историю.

Вот почему так важно знать математику!



Вавилонские ладони

Эту задачу придумали в Древнем Вавилоне примерно за 2 тысячи лет до нашей эры:

Длина и $\frac{1}{4}$ ширины вместе составляют 7 ладоней, а длина и ширина вместе – 10 ладоней.

Сколько ладоней составляют длина и ширина в отдельности?



Принято считать, что ширина не больше длины. Также из условия задачи следует, что ширина в ладонях кратна 4.

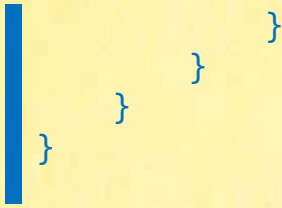
Эти два условия можно без труда перевести на язык *Cu++*:

```
#include <iostream>
#include<windows.h>

using namespace std;

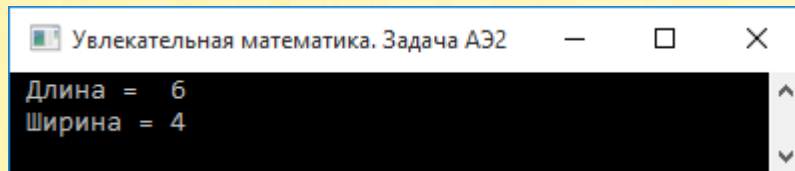
int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика. Задача АЭ2");

    for (int length = 1; ; ++length)
    {
        for (int width = 0; width <= length; width += 4)
        {
            if (length + width / 4 == 7 &&
                length + width == 10)
            {
                cout << " Длина = " << length << endl;
                cout << " Ширина = " << width << endl;
                cout << endl;
            }
        }
    }
    return 0;
}
```



Так как мы не знаем, чему равна длина, то перебираем значения в бесконечном цикле `for`. Ширина не больше длины, поэтому вложенный цикл конечный. Как только условие задачи выполнится, мы печатаем ответ на экране и завершаем программу оператором `return`.

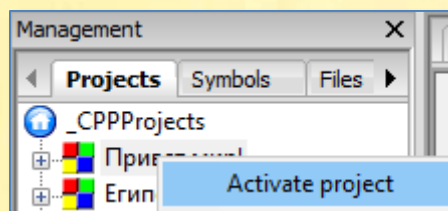
Итак, длина составляла 6 ладоней, а ширина – 4:



Впрочем, нетрудно сообразить, что $\frac{3}{4}$ ширины равны $10 - 7 = 3$ ладони, поэтому вся ширина равна 4 ладоням.

Полезный совет.

По умолчанию активным становится **новый** проект. Если вы хотите сделать **активным** другой проект из группы, то вызовите на его названии контекстное меню и щёлкните по строчке **Activate project**:



Индийские пчёлы



А эту задачу придумали в Древней Индии в то же время, что и задачу про длину и ширину в Древнем Вавилоне:

Пчёлы числом, равным квадратному корню из половины их числа во всём рое, сели на куст жасмина, $8/9$ пчёл полетели назад к рое. И только одна пчела из того же роя кружилась над цветком лотоса, привлечённая жужжанием подруги, неосторожно угодившей в ловушку сладко благоухающего цветка.

Сколько всего пчёл было в рое?

Так число пчёл – всегда **целое**, то в рое должно быть столько пчёл, чтобы их число нацело делилось и на 2 и на 9, то есть на 18. Этого наблюдения нам вполне достаточно, чтобы решить задачу:

```
#include <iostream>
#include<windows.h>
#include<math.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика. Задача АЭЗ");
}
```

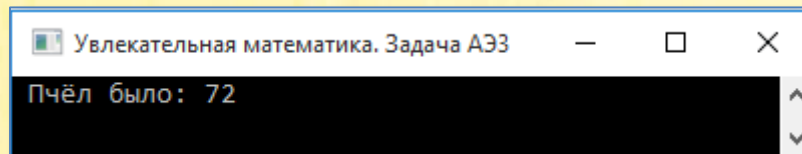
```

for (double bee = 0; ; bee += 18)
{
    double sqrtbee = sqrt(bee/2);
    if (sqrtbee + bee*8/9 + 2 == bee)
    {
        cout << " Пчёл было: " << bee << endl;
        cout << endl;
        break;
    }
}
return 0;
}

```

Для извлечения квадратного корня нам нужна функция **sqrt** из математической библиотеки.

Пчёл в рое было 72:



Алгебраическое решение, приведённое в книге *Увлекательная математика*, значительно сложнее переборного!

Как указывает автор этой книги, 4000 лет назад математика была своеобразным видом спорта. Устраивались состязания по решению сложных задач, на которых присутствовали многочисленные зрители.

Пчелиная задача взята из учебного пособия по проведению математических конкурсов, причём в оригинале она изложена в *стихотворной* форме.

Китайские кролики и фазаны



Одна из самых известных исторических занимательных задач – про кроликов и фазанов – была известна ещё в Древнем Китае. В трактате *Киу-чанг – Девять отделов арифметики*, - написанном в 2637 году до нашей эры, имеется такая задача:

В клетке находится неизвестное число фазанов и кроликов. Известно только, что общее число голов – 35 и число ног – 94.

Требуется узнать число фазанов и число кроликов.

На сайте <http://festival.1september.ru/articles/569698/> вы найдёте эту же задачу с подробным разбором её решения:

Сегодня на уроке мы вновь встретимся с вами с хорошо известной вам задачей про фазанов и кроликов (задача выводится на доску “В клетке находятся фазаны и кролики. Известно, что у них 35 голов и 94 ноги. Узнайте число фазанов и число кроликов”), но если раньше мы её решали арифметическим способом, то сегодня будем её решать с помощью уравнений и даже системы уравнений.

Поскольку ни число голов, ни число ног в программе не изменяется, то мы сохраним их в **константах**: HEADS - число *голов* и LEGS - число *ног*:

```
//общее число голов:  
const int HEADS = 35;  
//общее число ног:  
const int LEGS = 94;
```

Максимальное число **кроликов** (*Rabbits*) находим делением общего числа ног на 4 (у кролика 4 ноги, или лапы):

```
//макс. число кроликов:  
int maxRabbits = LEGS / 4;
```

Минимальное число кроликов равняется нулю – тогда все головы и ноги принадлежат только фазанам. Теперь мы должны перебрать все возможные числа для поголовья кроликов в цикле *for*. Ясно, что переменная цикла *i* должна изменяться в диапазоне $0..maxRabbits$. Число **фазанов** (*Pheasants*) мы найдём, вычтя из общего числа голов число кроликов, то есть *i*.

У *i* кроликов $i*4$ ноги, а у *nPheasants* фазанов - $nPheasants*2$ ног. Когда сумма ног совпадёт с требуемой (*LEGS*), мы печатаем решение.

В целом исходный код программы может быть таким:

```
#include <iostream>  
#include<windows.h>  
  
using namespace std;  
  
int main()  
{  
    SetConsoleOutputCP(1251);  
    system("title Наука и жизнь 1963-03-38-6");  
  
    //общее число голов:
```

```

const int HEADS = 35;
//общее число ног:
const int LEGS = 94;

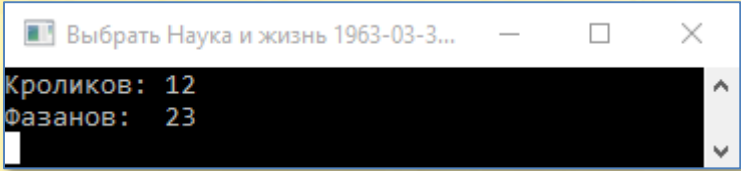
//макс. число кроликов:
int maxRabbits = LEGS / 4;

//решаем задачу:
for (int i = 0; i <= maxRabbits; ++i)
{
    //число фазанов:
    int nPheasants = HEADS - i;
    if (i * 4 + nPheasants * 2 == LEGS)
    {
        cout << "Кроликов: " << i << endl;
        cout << "Фазанов:  " << nPheasants;
    }
}
cout << endl;

return 0;
}

```

Запускаем программу и читаем ответ на задачу: кроликов было 12, а фазанов – 23.



Выбрать Наука и жизнь 1963-03-3... — □ ×

```

Кроликов: 12
Фазанов: 23

```

Эту древнюю задачу до сих пор решают в средней школе:

У фазанов и кроликов 62 ноги и 19 голов.

Сколько фазанов и кроликов?

Достаточно изменить значения констант – и ответ получен:

```

int main()
{
    SetConsoleOutputCP(1251);
    system("title Кролики и фазаны");

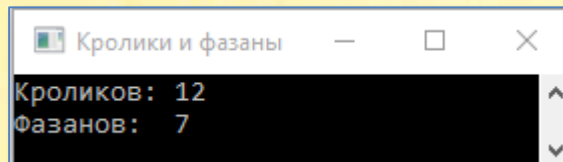
    //общее число голов:
    const int HEADS = 19;
    //общее число ног:
    const int LEGS = 62;

    //макс. число кроликов:
    int maxRabbits = LEGS / 4;

    //решаем задачу:
    for (int i = 0; i <= maxRabbits; ++i)
    {
        //число фазанов:
        int nPheasants = HEADS - i;
        if (i * 4 + nPheasants * 2 == LEGS)
        {
            cout << "Кроликов: " << i << endl;
            cout << "Фазанов:  " << nPheasants;
        }
    }
    cout << endl;

    return 0;
}

```



```

Кролики и фазаны
Кроликов: 12
Фазанов: 7

```

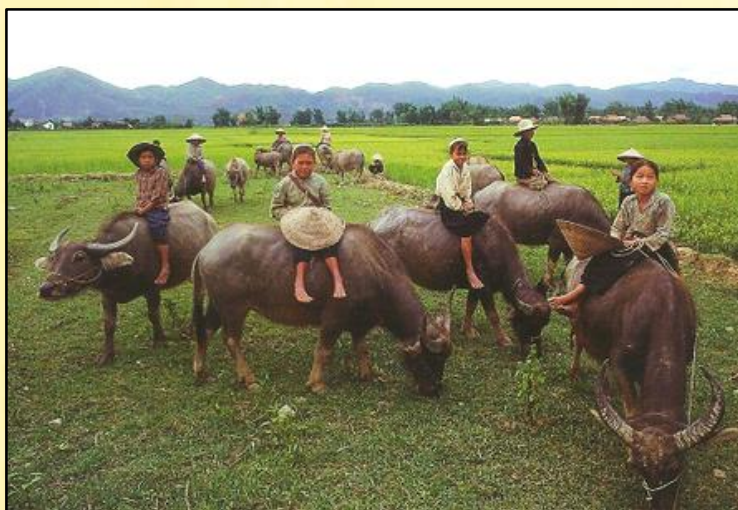
А вот как решали задачу про кроликов и фазанов сами китайцы:

Если бы в клетке были одни фазаны, то число ног было бы 70, а не 94. Следовательно, 24 лишних ноги принадлежат кроликам, по две на каждого. Кроликов – 12, фазанов – 23.

Иногда хочется быть древним китайцем!



Вьетнамские буйволы



Вот очень старая вьетнамская задача, которую старики-рисоводы любят задавать подрастающему поколению:

Для кормления 100 буйволов заготовили 100 охапок сена.

Стоящий молодой буйвол съедает 5 охапок сена.

Лежащий молодой буйвол съедает 3 охапки сена.

Старые буйволы втроем съедают 1 охапку сена.

Сколько молодых буйволов стоят, сколько лежат и сколько буйволов старых?

Решение задачи с буйволами не очень сильно отличается от решения кроличье-фазаньей.

Понятно, что буйволов каждого вида не меньше нуля и не больше – (100 поделить на число съедаемых охапок сена). Некоторую нервозность вносят старые буйволы, которых следует считать тройками.

Когда общее число буйволов и съедаемых охапок сена будет по 100, мы печатаем ответ:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика 2");

    for (int buffaloS = 0; buffaloS <= 100/5; ++buffaloS)
        for (int buffaloL= 0; buffaloL <= 100/3; ++buffaloL)
            for (int buffaloO= 0; buffaloO <= 100; buffaloO += 3)
                {
                    if (buffaloS + buffaloL + buffaloO == 100 &&
                        buffaloS*5 + buffaloL*3 + buffaloO/3 == 100)
                        {
                            cout << "Стоящих молодых буйволов : " << buffaloS << endl;
                            cout << "Лежащих молодых буйволов : " << buffaloL << endl;
                            cout << "Старых буйволов          : " << buffaloO << endl;
                            cout << endl;
                        }
                }
}
```



```
return 0;
```

```
}
```

Задача имеет **4 решения**, что вполне могло поставить вьетнамских недорослей в тупик.

```
Увлекательная математика 2
Состоящих молодых буйволов : 0
Лежащих молодых буйволов : 25
Старых буйволов : 75

Состоящих молодых буйволов : 4
Лежащих молодых буйволов : 18
Старых буйволов : 78

Состоящих молодых буйволов : 8
Лежащих молодых буйволов : 11
Старых буйволов : 81

Состоящих молодых буйволов : 12
Лежащих молодых буйволов : 4
Старых буйволов : 84
```

Индийские обезьяны



Две очень старые индийские задачи про обезьян. Подобные задачи были известны в Индии ещё в XII веке.

Сколько обезьян в стаде, если квадрат одной пятой их без 3 скрылось в пещере, а одна залезла на дерево?

Эта задача легко решается с помощью квадратного уравнения. Но это в Индии – там тепло. А нам сподручнее решить задачу с помощью перебора:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д10");

    for (int monkey = 0; ; ++monkey)
    {
        int m5 = monkey / 5 - 3;
        if (m5 > 0 && m5 * m5 + 1 == monkey)
        {
            cout << " Обезьян было: " << monkey << endl;
            cout << endl;
            break;
        }
    }

    return 0;
}
```

Условие

`m5 * m5 + 1 == monkey`

следует из текста задачи и представляет собой как раз квадратное уравнение. Оно имеет 2 корня – 5 и 50.

Условие

$m5 > 0$

игнорирует первый корень, поскольку число обезьян, скрывшихся в пещере, не может быть отрицательным. Остаётся второй корень – 50:

```
Математический фольклор. Задача Д10
Обезьян было: 50
```

Индийские обезьяны 2



Квадрат восьмой части стада обезьян играют в роще, а остальные 12 – на горке.

Сколько обезьян в стаде?

Поскольку задача опять решается с помощью квадратного уравнения, то нам нужно найти **оба** корня. Для этого в бесконечном цикле *for* мы считаем найденные решения:

```

#include <iostream>
#include<windows.h>

using namespace std;

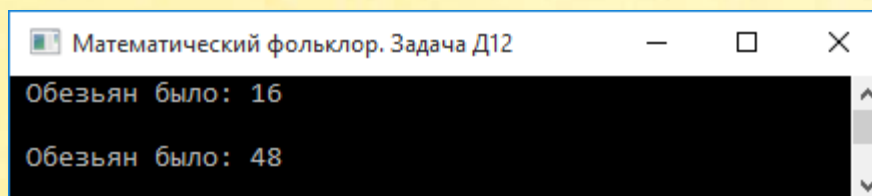
int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д12");

    //число решений:
    int var = 0;
    for (int monkey = 0; ; monkey += 8)
    {
        int m8 = monkey / 8;
        if (monkey - m8 * m8 == 12)
        {
            cout << " Обезьян было: " << monkey << endl;
            cout << endl;
            if (++var == 2)
                break;
        }
    }

    return 0;
}

```

На этот раз оба ответа правильные:



```

Математический фольклор. Задача Д12
Обезьян было: 16
Обезьян было: 48

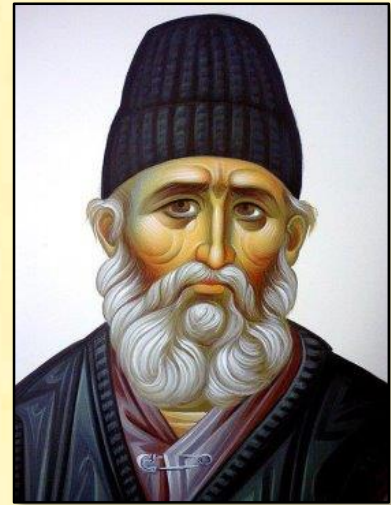
```

Жизнь Демохара

Множество занимательных задач связано с вычислением **возраста**. Одна из первых появилась в Греции в конце IV века:

Демохар четверть жизни прожил мальчиком, пятую часть – юношей, одну треть – зрелым мужчиной и 13 лет пожилым.

Сколько лет прожил Демохар?



Демохар прожил не меньше 13 лет (наверняка гораздо дольше, но перебор всё равно будет небольшим, поэтому мы можем удовлетвориться и этим явно заниженным возрастом). Так как в условии присутствует деление, то возраст лучше хранить в переменной типа *double*:

```
#include <iostream>
#include<windows.h>

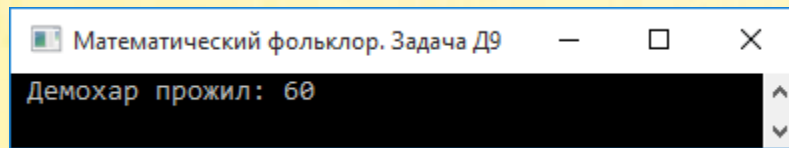
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д9");

    for (double let = 13.0; ; ++let)
    {
        if (let/4 + let/5 + let/3 + 13 == let)
        {
            cout << " Демохар прожил: " << let;
            cout << endl;
            break;
        }
    }
}
```

```
}  
    return 0;  
}
```

Демохар прожил 60 лет:



Если мы будем считать, что результат деления всегда **целое** число, то возраст Демохара должен нацело делиться на 4, 5 и 3. Минимальное число, удовлетворяющее этим требованиям, - 60. Следующее число – 120 – явно не по Демохару.

Греческие мешконосы

Эта задача приписывается Евклиду. Ей около 2300 лет!

Нагруженные осёл и мул идут очень медленно. Осёл жалуется на непосильную ношу, а мул отвечает:

- Что ты жалуешься? Если я возьму один твой мешок, то моя ноша станет в 2 раза тяжелее твоей. А если ты возьмёшь один мой мешок, то наши ноши будут равны.

По сколько мешков несли осёл и мул?



Эту задачу можно найти в книге *Увлекательная математика. Античные этюды. Задача 7:*

Однажды мула и осла нагрузили зерном. По дороге мул сказал ослу:

- Если бы ты уступил мне одну меру своего груза, то я нёс бы вдвое больше зерна, чем ты. А если бы я уступил тебе одну меру своего груза, то мы оба несли бы зерна поровну.

По сколько мер зерна нёс мул и сколько - осёл?

И осёл, и мул несли не менее 1 мешка (даже не меньше двух). Здравый смысл нам подсказывает, что мул нёс не более 100 мешков (наверняка гораздо меньше). Сколько мешков нёс осёл, нам знать необязательно, потому что мы будем добавлять ему мешки в бесконечном цикле *for*:

```
#include <iostream>
#include<windows.h>

using namespace std;

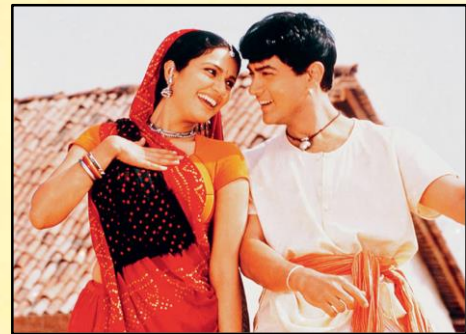
int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д8");

    for (int osel = 1; ; ++osel)
        for (int mul = 1; mul <= 100; ++mul)
            {
                bool uslovie1 = mul + 1 == 2 * (osel - 1);
                bool uslovie2 = mul - 1 == osel + 1;
                if (uslovie1 && uslovie2)
                    {
                        cout << " Осёл нёс: " << osel << endl;
                        cout << " Мул нёс: " << mul << endl;
                        cout << endl;
                        return 0;
                    }
            }
}
```

Ответ на задачу:

```
Математический фольклор. Задача Д8
Осёл нёс: 5
Мул нёс: 7
```

Индийское число



Условие задачи:

Если некоторое число умножить на 5, от произведения отнять его треть, остаток разделить на 10 и к полученному числу прибавить последовательно одну треть, одну вторую и одну четвертую первоначального числа, то получится 68.

Какое это число?

Мы можем только посочувствовать древним индийцам, у которых не было компьютеров, чтобы решать такие задачи.

А для компьютера эта задача очень простая – так же, как и для нас. Главное - правильно записать её условие:

```
#include <iostream>
#include <windows.h>
```



```

using namespace std;

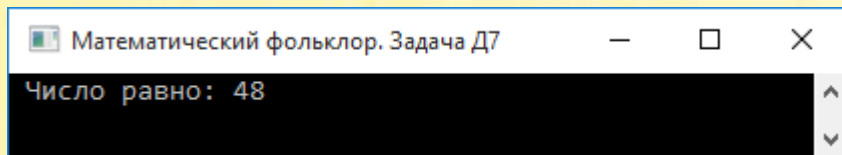
int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д7");

    for (int num = 1; ; ++num)
    {
        if ((num * 5 - num*5/3)/10 + num/3 + num/2 + num/4 == 68)
        {
            cout << " Число равно: " << num << endl;
            cout << endl;
            break;
        }
    }

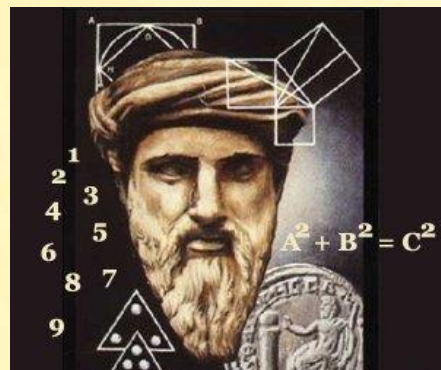
    return 0;
}

```

Мы могли бы предположить, что искомое число должно делиться нацело на 2, 3 и 4 и тем значительно сократить перебор, но в данном случае этого не требуется:



Пифагорейское число



Знаменитого Пифагора спросили:

- Сколько учеников посещают твою школу и слушают твои беседы?

Пифагор ответил:

- Половина моих учеников изучает математику, четверть – музыку, седьмая часть проводит время в молчаливом размышлении, остальную часть составляют 3 ученика.

Сколько учеников было у Пифагора?

Эта задача напечатана и в книге Увлекательная математика. Античные этюды. Задача 5. Пифагор Самосский (ок. 508-501 гг. до н.э.):

Поликрат (известный из баллады Шиллера тиран с острова Самос) однажды спросил на пиру у Пифагора, сколько у того учеников.

- Охотно скажу тебе, о Поликрат, - отвечал Пифагор. - Половина моих учеников изучает прекрасную математику, четверть исследует тайны вечной природы, седьмая часть молча упражняет силу духа, храня в сердце учение. Добавь ещё к ним трёх юношей, из которых Теон превосходит прочих своими способностями.

Сколько учеников было у Пифагора?

Так как число учеников нужно делить на 2, 4 и 7, то для переменной бесконечного цикла *for* следует выбрать тип *double*. Условие прекращения цикла легко выводится из текста задачи:

```
#include <iostream>
#include<windows.h>

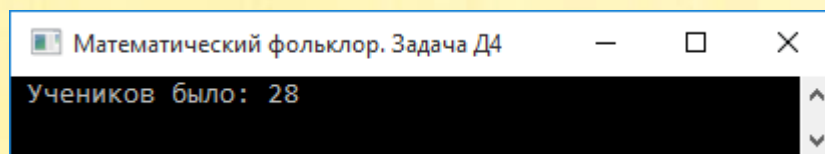
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д4");

    for (double uchenik = 1; ; ++uchenik)
    {
        if (uchenik/2 + uchenik/4 + uchenik/7 + 3 == uchenik)
        {
            cout << " Учеников было: " << uchenik << endl;
            cout << endl;
            break;
        }
    }

    return 0;
}
```

У Пифагора было 28 учеников:



Задача легко решается в уме, если учесть, что число учеников должно быть кратно 4 и 7.

Индийский храм



Из четырёх посетителей храма второй дал в 2 раза больше монет, чем первый, третий – в 3 раза больше монет, чем второй, а четвёртый – в 4 раза больше монет, чем третий. Всего было дано 132 монеты.

Сколько монет дал первый?

Мы будем увеличивать в бесконечном цикле *for* число монет первого посетителя храма. Число монет остальных посетителей легко найти с помощью умножения. Когда общая сумма достигнет 132, цикл завершается:

```
#include <iostream>
#include<windows.h>

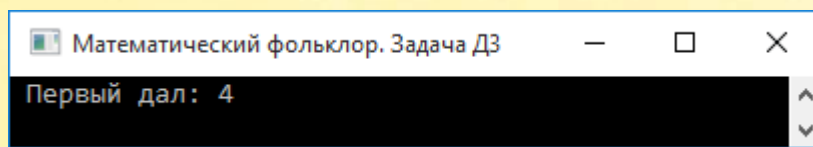
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д3");

    for (int monet = 1; ; ++monet)
    {
        if (monet + monet * 2 + monet * 2 * 3 +
            monet * 2 * 3 * 4 == 132)
        {
```

```
        cout << " Первый дал: " << monet << endl;  
        cout << endl;  
        break;  
    }  
}  
  
return 0;  
}
```

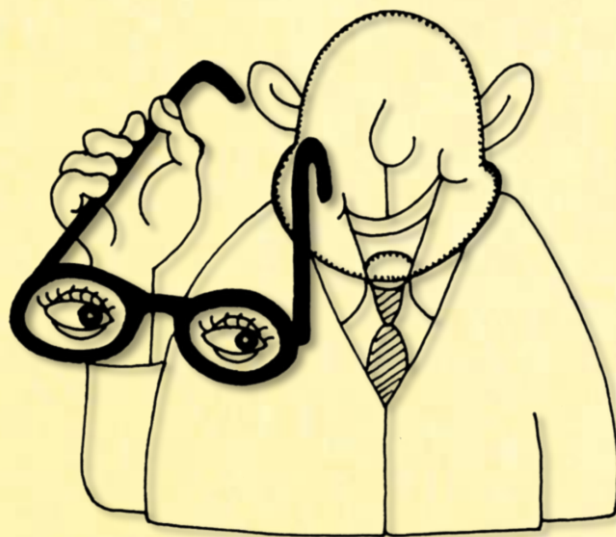
Первый посетитель дал **4 монеты**:



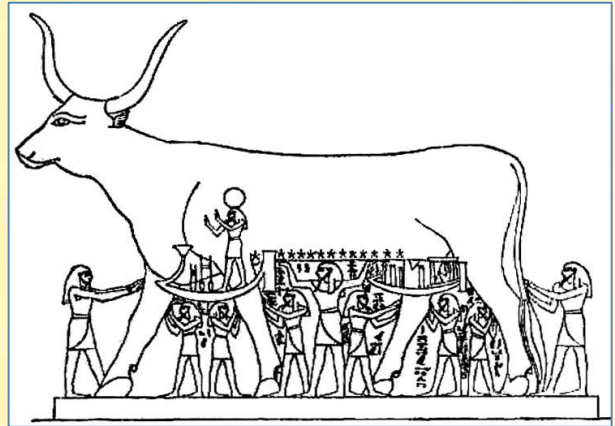
В Древней Греции такие задачи решали **методом приведения к единице**. Он напоминает наш перебор, но умнее.

Действительно, если первый посетитель даст одну монету, то общая сумма составит 33 монеты. Это в 4 раза меньше требуемой. Следовательно, во столько же раз больше первый посетитель должен был дать монет. То есть 4.

Как хорошо, что у древних греков не было компьютеров!



Египетские коровы



В этом проекте мы решим одну задачу из папируса Ринда (Райнда). Возраст папируса оценивается в 3700 лет. Он был найден англичанином Риндом в конце XIX века.

Но папирус – только часть более древнего математического труда третьего тысячелетия до нашей эры:

Некий математик насчитал на выгоне 70 коров.

- Какую долю от всего стада составляют эти коровы? – спросил математик у пастуха.

- Я выгнал пастись две трети от трети всего стада, - отвечал пастух.

Сколько голов скота насчитывается во всём стаде?

Так как коровы паслись не консервированные, а цельные, то число голов в стаде должно делиться на 9.

Задача легко решается и без перебора, поэтому мы применяем компьютер только из почтения к древности этой математической задачи:

```
#include <iostream>
#include<windows.h>
```

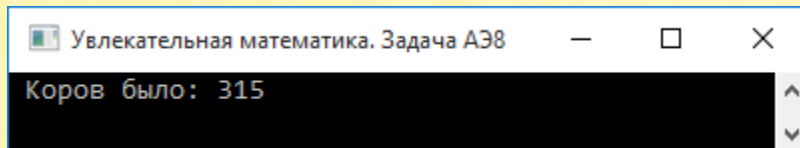
```
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика. Задача АЭ8");

    for (int cow = 0; ; cow += 9)
    {
        if (cow*2/3*1/3 == 70)
        {
            cout << " Коров было: " << cow << endl;
            cout << endl;
            break;
        }
    }

    return 0;
}
```

А коров было много:



Римские адвокаты



Крючкотворная римская задача первого века до нашей эры:

Адвокаты в Древнем Риме имели обыкновение задавать друг другу задачи. Одна из таких задач гласит:

Некая вдова должна разделить оставшееся после смерти мужа наследство в размере 3500 динариев с ещё не родившимся ребёнком. По римским законам, если родится сын, то мать получает половину причитающейся ему доли, а в случае рождения дочери мать получает вдвое больше неё. У вдовы родились близнецы – сын и дочь.

Как разделить наследство, чтобы все требования закона были соблюдены?

Мы полагаем, что задача решается в **целых** числах. Тогда легко найти долю вдовы в бесконечном цикле **for**. Совершенно очевидно, что она не может получить больше 3500 динариев (и даже значительно меньше). Тогда сын получает вдвое больше, а дочь вдове меньше неё:

```
#include <iostream>
#include <windows.h>

using namespace std;
```



```

int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика. Задача АЭ13");

    for (int vdova = 0; vdova <= 3500; ++vdova)
    {
        int syn = 2 * vdova;
        int dotch = vdova / 2;
        if (vdova + syn + dotch == 3500)
        {
            cout << " Вдове:  " << vdova << endl;
            cout << " Сыну:    " << syn << endl;
            cout << " Дочери: " << dotch << endl;
            cout << endl;
            break;
        }
    }

    return 0;
}

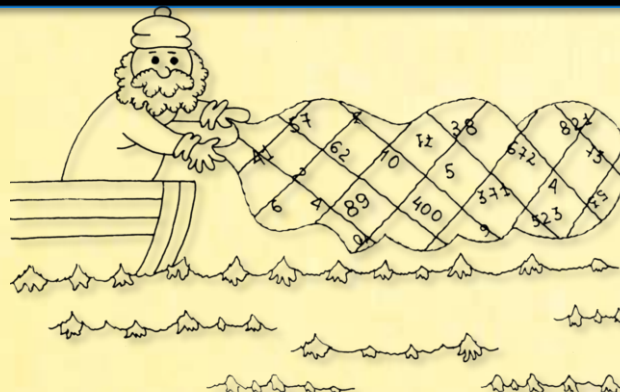
```

Всё – мы разделили наследство по-честному, то есть по древнеримским законам:

```

Увлекательная математика. Задача АЭ13
Вдове: 1000
Сыну: 2000
Дочери: 500

```



Диофантово число



Задача древнегреческого математика Диофанта Александрийского, жившего в третьем веке:

По двум данным числам 200 и 5 найти третье число, которое, если его умножить на одно из них, даёт полный квадрат, а если его умножить на другое число, даёт квадратный корень из этого квадрата.

Мы можем найти заданное число перебором в бесконечном цикле *for*, начиная с единицы. Полный квадрат мы найдём умножением 200 (но не 5!) на текущее значение переменной цикла. Для определения принадлежности произведения к полным квадратам мы напишем функцию `isQuadrat`:

```
#include <iostream>
#include<windows.h>
#include<math.h>

using namespace std;

//ОПРЕДЕЛЯЕМ ЯВЛЯЕТСЯ ЛИ ЗАДАННОЕ ЧИСЛО
//ПОЛНЫМ КВАДРАТОМ
bool isQuadrat(long num)
{
```

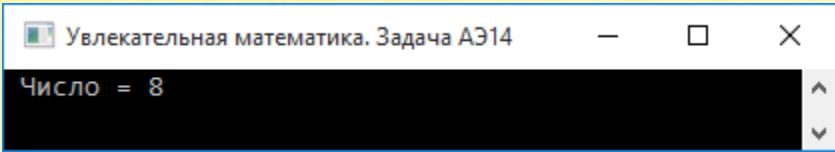
```
    long r = (long)sqrt(num);  
    return r * r == num;  
}
```

Понятно, что в этой задаче нам не обойтись без математической библиотеки *C++*.

Главная функция программы короткая и простая:

```
int main()  
{  
    SetConsoleOutputCP(1251);  
    system("title Увлекательная математика. Задача АЭ14");  
  
    for (int num = 1; ; ++num)  
    {  
        if (isQuadrat(num*200) && sqrt(num*200) == 5*num)  
        {  
            cout << " Число = " << num << endl;  
            cout << endl;  
            break;  
        }  
    }  
  
    return 0;  
}
```

Искомое число равно 8:



```
Увлекательная математика. Задача АЭ14  
Число = 8
```

Годится также число 0, но оно явно не древнегреческое.

Если не копировать полностью требование задачи, то условие можно записать проще:

```
if (5*num == sqrt(num*200))
```

Арабские голуби



Задача из арабских сказок *1001 ночь* (ночь 458), которые были написаны много столетий назад:

Стая голубей подлетела к высокому дереву. Часть голубей села на ветвях, а другая расположилась под деревом. Сидевшие на ветвях голуби говорят расположившимся внизу:

- Если бы один из вас взлетел к нам, то вас стало бы втрое меньше, чем нас всех вместе, а если бы один из нас слетел к вам, то нас с вами стало бы поровну.

Сколько голубей сидело на ветвях и сколько под деревом?

Поскольку никаких намёков на число голубей мы не имеем, то только первый цикл *for* может быть бесконечным. Во вложенном цикле мы полагаем, что голубей было меньше тысячи:

```

#include <iostream>
#include<windows.h>

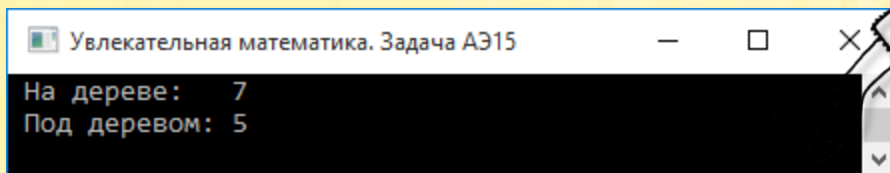
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика. Задача АЭ15");

    for (double na = 0; ; ++na)
        for (double pod = 0; pod < 1000; ++pod)
        {
            if (pod - 1 == (na + pod)/3 &&
                na - 1 == pod + 1)
            {
                cout << " На дереве:   " << na << endl;
                cout << " Под деревом: " << pod << endl;
                cout << endl;
                return 0;
            }
        }
}

```

На самом-то деле голубей было совсем немного:



Персидские яблоки



Задача из старинной персидской легенды *История Морадбальса*, включённой в сборник сказок *1001 ночь*. В ней мудрец задаёт молодой девушке такую задачу:

Одна женщина отправилась в сад собрать яблоки. Чтобы выйти из сада, ей нужно было пройти через 4 двери, у каждой из которых стоял стражник. Стражнику у первых дверей женщина отдала половину сорванных ею яблок. Дойдя до второго стражника, женщина отдала ему половину оставшихся яблок. Так же она поступила и с третьим стражником; а когда она поделилась яблоками со стражником у четвёртых дверей, то у неё осталось 10 яблок.

Сколько яблок она собрала в саду?

Поучительная задача, актуальная и в наши дни: чтобы пройти через двери, нужно делать взносы (или вклады).

Мы перебираем число яблок в бесконечном цикле *for*, делимся ими со стражниками до тех пор, пока в результате не останется ровно десяток яблок:

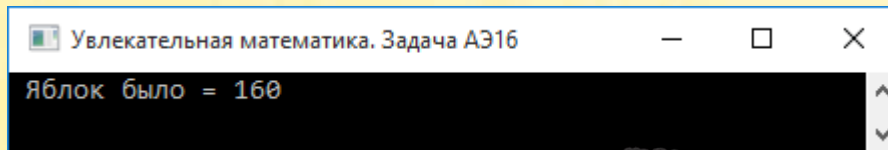
```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
```

```
system("title Увлекательная математика. Задача АЭ16");  
  
for (int apple = 1; ; ++apple)  
{  
    int ost = apple;  
    for (int n = 1; n <= 4; ++n)  
    {  
        ost /= 2;  
    }  
    if (ost == 10)  
    {  
        cout << " Яблок было = " << apple << endl;  
        cout << endl;  
        break;  
    }  
}  
  
return 0;  
}
```

А девушка, видимо, не в первый раз промыляла яблоками, так как нарвала их с большим запасом:



Кахунский папирус



Очень старая задача из *Кахунского папируса*:

Отношение чисел равно $2 : 1\frac{1}{2}$, сумма квадратов – 400.

Найти эти числа.

Решение задачи очень простое. Главное – не запутаться в дробях:

```
#include <iostream>
#include<windows.h>

using namespace std;

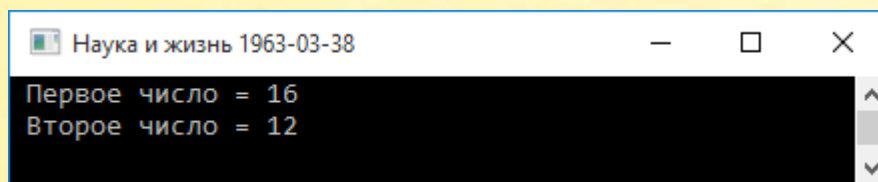
int main()
{
    SetConsoleOutputCP(1251);
    system("title Наука и жизнь 1963-03-38");

    for (int num1 = 1; ; ++num1)
    {
        double num2 = num1/2*3/2;
        if (num1 * num1 + num2 * num2 == 400)
        {
            cout << " Первое число = " << num1 << endl;
            cout << " Второе число = " << num2 << endl;
        }
    }
}
```



```
        cout << endl;  
        break;  
    }  
}  
  
return 0;  
}
```

Задача показывает, что древние математики не хуже нашего умели вычислять дроби и квадраты чисел:



```
Наука и жизнь 1963-03-38  
Первое число = 16  
Второе число = 12
```

Берлинский папирус



Ещё одна древневавилонская задача. На этот раз из *Берлинского папируса*:

Если тебе будет сказано: разделить 100 квадратных локтей на 2 неизвестные части и $\frac{3}{4}$ стороны одной взять за сторону другой, дай мне каждую из неизвестных частей (иначе говоря, нужно разбить площадь в 100 квадратных локтей на 2 квадрата, стороны которых относились бы как 1 : $\frac{3}{4}$).

Так как стороны квадратов выражаются **целыми** числами, то длина стороны меньшего квадрата должна быть кратна трём:

```
#include <iostream>
#include<windows.h>

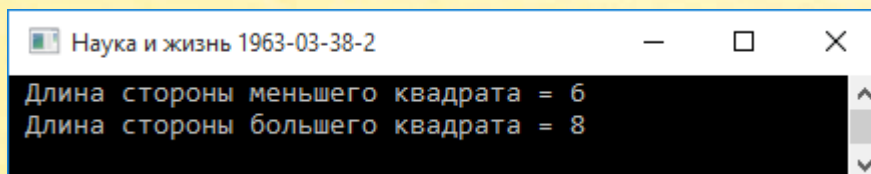
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Наука и жизнь 1963-03-38-2");

    for (double len = 0; ; len += 3)
    {
        //длина второй стороны:
        double len2 = len * 4 / 3;
        if ( len * len + len2 * len2 == 100)
        {
            cout << " Длина стороны меньшего квадрата = " << len << endl;
            cout << " Длина стороны большего квадрата = " << len2 << endl;
            cout << endl;
            break;
        }
    }

    return 0;
}
```

Ответ на задачу:



```
Наука и жизнь 1963-03-38-2
Длина стороны меньшего квадрата = 6
Длина стороны большего квадрата = 8
```

Индийские квадраты



Индийская задача VIII века из «Бахшалийской» рукописной арифметики:

Найти число, которое от прибавления 5 или отнятия 11 обращается в полный квадрат.

Искомое число не меньше 11, а верхнюю границу мы установим в сотню:

```
#include <iostream>
#include<windows.h>
#include<math.h>

using namespace std;

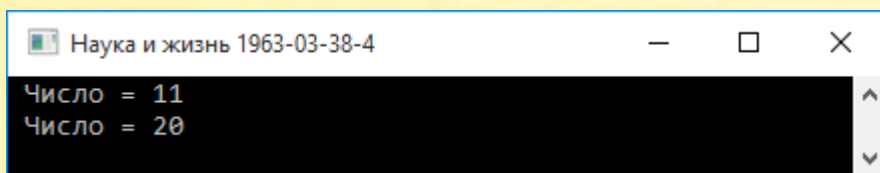
//ОПРЕДЕЛЯЕМ ЯВЛЯЕТСЯ ЛИ ЗАДАННОЕ ЧИСЛО
//ПОЛНЫМ КВАДРАТОМ
bool isQuadrat(long num)
{
    long r = (long)sqrt(num);
    return r * r == num;
}

int main()
{
    SetConsoleOutputCP(1251);
    system("title Наука и жизнь 1963-03-38-4");
```

```
for (int num = 11; num < 100; ++num)
{
    int n5 = num + 5;
    int n11 = num - 11;
    if (isQuadrat(n5) && isQuadrat(n11))
    {
        cout << " Число = " << num << endl;
    }
}
cout << endl;

return 0;
}
```

Мы нашли 2 числа, удовлетворяющих условиям задачи:



```
Наука и жизнь 1963-03-38-4
Число = 11
Число = 20
```

При **num = 11** мы получаем квадраты 16 и 0, а при **num = 20** – 25 и 9.



СТАРЫЕ ЗАДАЧИ

Время неумолимо отсчитывает за веком век, занимательнее задачи становятся всё интереснее, изощрённее и многообразнее. Некоторые из них благополучно пережили века и до сих пор будоражат воображение и привлекают внимание любителей поломать голову.

В этой главе мы решим пару десятков любопытных задачек со всего света.

Чисто американская задача



Американцы любят деньги – доллары и центы. И почти все американские задачи только про деньги. Давайте поможем им и решим старую (судя по ценам) задачу про деньги.

Мужчина потратил в магазине половину денег, которые имел при себе. Он заметил, что в кармане у него осталось столько центов, сколько долларов было до магазина, а долларов осталось половина от того, сколько центов он имел перед этим.

Сколько денег было у мужчины сначала?

Так как центов могло остаться от 0 до 99, то первоначально и долларов было от 0 до 99.

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д45");
```

Решаем задачу с помощью двух вложенных циклов *for*. Во внешнем цикле мы изменяем число долларов в заданном диапазоне, а во вложенном цикле - число центов:

```
for (int d = 0; d <= 99; ++d)
    for (int c = 0; c <= 99; ++c)
    {
```

По условию задачи, у американского мужчины осталось столько центов, сколько было долларов, а долларов – половина от прежних центов:

```
int c1 = d;
int d1 = c / 2;
```

После лёгкой прогулки по магазину у него осталась половина суммы, что мы дальше и проверяем. Если книжное условие выполняется, мы печатаем ответ на задачу:

```
        if (100 * d + c == 2 * (100 * d1 + c1))
        {
            cout << "\n Долларов было: " << d <<
                "\n Центов было: " << c;
            cout << "\n Долларов стало: " << d1 <<
                "\n Центов стало: " << c1;
            cout << endl;
        }
    }
    return 0;
}
```

Довольно забавно, но задача имеет 2 решения:

```
Математический фольклор. Задача Д45

Долларов было: 0
Центов было: 0
Долларов стало: 0
Центов стало: 0

Долларов было: 99
Центов было: 98
Долларов стало: 49
Центов стало: 99
```

Первый ответ более правдоподобен: но в книге приводится **второй** ответ.

Чисто французская задача



Французы мало уступают американцам по части любви к деньгам, но уже к своим. В те далёкие времена это были франки и сантимы.

Мсье Метивье с женой решили отпраздновать годовщину своей свадьбы в ресторане.

Уходя, мсье Метивье заплатил по счёту и заметил, что у него осталась одна пятая денег, которые были с собой. Причём сантимов осталось столько, сколько франков было вначале (1 франк = 100 сантимов), а оставшихся франков было в 5 раз меньше, чем сантимов вначале.

Какую сумму мсье Метивье заплатил в ресторане?

Что касается арифметической подоплёки предложенной нам задачи, то достаточно выгодно обменять доллары и центы на франки и сантимы, а также учесть дефицит французского бюджета – и задача решена!

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д44");

    for (int f = 0; f <= 99; ++f)
        for (int c = 0; c <= 99; ++c)
            {
                int c1 = f;
                int f1 = c / 5;
                if (100 * f + c == 5 * (100 * f1 + c1))
                    {
                        cout << "\n Франков было: " << f <<
                            "\n Сантимов было: " + c;
                        cout << "\n Франков стало: " << f1 <<
                            "\n Сантимов стало: " << c1;
                        int zaplatil = 4*(100 * f1 + c1);
                        cout << "\n Мсье Метивье заплатил: " <<
                            (zaplatil/100) << " фр. и " <<
                            (zaplatil - zaplatil / 100 * 100) << "с.";
                        cout << endl;
                    }
            }

    return 0;
}
```

Решение задачи слегка осложняется бухгалтерскими выкладками, но мы, умело сведя дебит с кредитом, тут же получаем расчёт счёта:

```
Математический фольклор. Задача Д44
Франков было: 0
Сантимов было:
Франков стало: 0
Сантимов стало: 0
Мсье Метивье заплатил: 0 фр. и 0с.

Франков было: 99:
Франков стало: 19
Сантимов стало: 99
Мсье Метивье заплатил: 79 фр. и 96с.
```

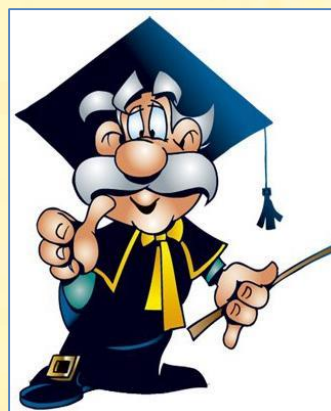
Поскольку мсье Метивье вряд ли осмелился бы пойти с женой в ресторан без франков и сантимов, то следует признать верным только **второй** ответ.

Репетитор

А вот русская задача вообще не про деньги, а, наоборот, про учёбу. И про то, как получить за это деньги.

В те далёкие времена, когда компьютеров ещё и в помине не было, школьники уже всю решали задачки по математике, что давалось им нелегко. Этот познавательный процесс хорошо описан Антоном Павловичем Чеховым в рассказе *Репетитор*.

Я надеюсь, что вы с удовольствием прочитаете этот рассказ Чехова (а за ним и многие другие) от начала до конца, поэтому перейдём непосредственно к **задаче**:



*Теперь по арифметике... Берите доску. Какая следующая задача?
Петя плюёт на доску и стирает рукавом. Учитель берёт задачник и диктует:*

- «Купец купил 138 арш. чёрного и синего сукна за 540 руб. Спрашивается, сколько аршин купил он того и другого, если синее стоило 5 руб. за аршин, а чёрное 3 руб.?»

Тяготы репетиторского труда мы пропускаем и читаем финал этой арифметической истории:

- И без алгебры решить можно, - говорит Удодов, протягивая руку к счётам и вздыхая. - Вот, извольте видеть...

Он щёлкает на счётах, и у него получается 75 и 63, что и нужно было.

Итак, ответ на задачу известен, и нам только и остаётся, что заменить старые счёты современными, то есть компьютером.

Обозначаем **длину** (*length*) сукна и **стоимость** (*cost*) соответствующими **константами**:

```
//общая длина сукна в аршинах:  
const int LENGTH = 138;  
//общая стоимость сукна в рублях:  
const int COST = 540;
```

На *COST* рублей можно купить не более **maxBlue** аршин **синего** сукна:

```
//макс. длина синего сукна:  
int maxBlue = COST/5;
```

Вы, конечно заметили, что эта задача принципиально ничем не отличается от старинной кроличье-фазаньей: заменив кроликов и фазанов **синим** и **чёрным** сукном, мы быстро выписываем весь переборный цикл **for**:

```
//решаем задачу:  
for (int i=0; i <= maxBlue; ++i)
```

```

{
    //длина чёрного сукна:
    int lenBlack = LENGTH - i;
    if (i*5 + lenBlack*3 == COST)
    {
        cout << "Синего сукна (аршин): " << i << endl;
        cout << "Чёрного сукна (аршин): " << lenBlack << endl;
    }
}

```

Нажимаем кнопку *Build and run* и получаем верный ответ:

```

Репетитор
Синего сукна (аршин): 63
Чёрного сукна (аршин): 75

```

Как видите, решение таких задач на компьютере – дело техники, а вот на счётах – это уже искусство...

Американские цыпочки

В очередной американской задаче мы снова встречаемся с долларами. На этот раз одинокими – без центов. Несколько скрашивают излишне меркантильную картину цыпочки, уточки и гусачки.

На вопрос, сколько стоит товар, продавец ответил:

- 3 цыплёнка и 1 утка стоят столько, сколько 2 гуся.

1 цыплёнок, 2 утки и 3 гуся вместе стоят 25 долларов. Причём каждый цыплёнок, утка и гусь стоят целое число долларов.



Сколько стоит каждая птица?

Из последнего равенства следует, что **цыплёнок** стоит не дороже 25 долларов, **утка** – не дороже 12 долларов и **гусь** – не дороже 8 долларов. Три вложенных цикла **for** – и мы узнаем цену американским цыпочкам:

```
#include <iostream>
#include<windows.h>

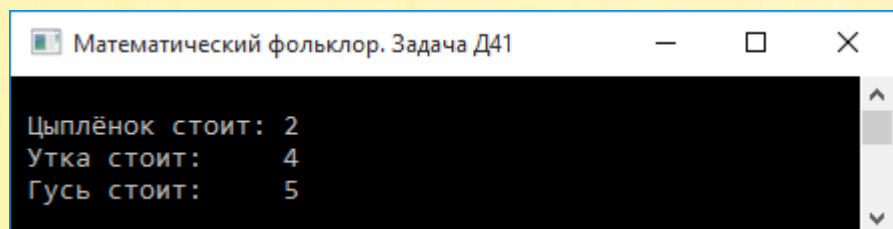
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д41");

    for (int cyplenok = 0; cyplenok <= 25; ++cyplenok)
        for (int gus = 0; gus <= 8; ++gus)
            for (int utka = 0; utka <= 12; ++utka)
            {
                if (3 * cyplenok + utka == 2 * gus &&
                    cyplenok + 2 * utka + 3 * gus == 25)
                {
                    cout << "\n Цыплёнок стоит: " << cyplenok <<
                        "\n Утка стоит:      " << utka <<
                        "\n Гусь стоит:      " << gus;
                    cout << endl;
                }
            }

    return 0;
}
```

А вот и цены:



```
Математический фольклор. Задача Д41
Цыплёнок стоит: 2
Утка стоит:      4
Гусь стоит:      5
```

Американское наследство



Вы будете смеяться, но снова американская задача про американские деньги:

Отец оставил сыновьям Чарлзу и Роберту 100 долларов.

Если одну треть части Чарлза вычесть из одной четверти части Роберта, то останется 11 долларов.

Сколько долларов получил каждый из братьев?

Обычно делёж наследства – процесс сложный и неприятный, но сотню долларов мы легко поделим – не поровну, но по-честному:

```
#include <iostream>
#include<windows.h>

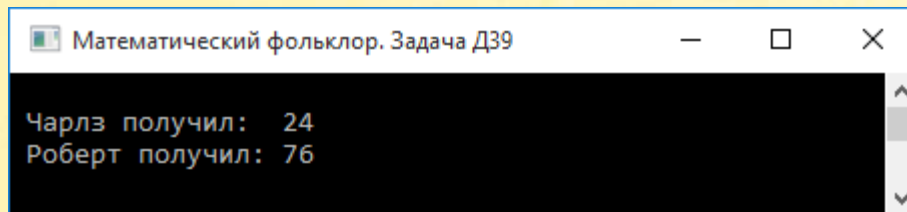
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д39");

    for (int Charles = 0; Charles <= 100; ++Charles)
    {
        int Robert = 100 - Charles;
```

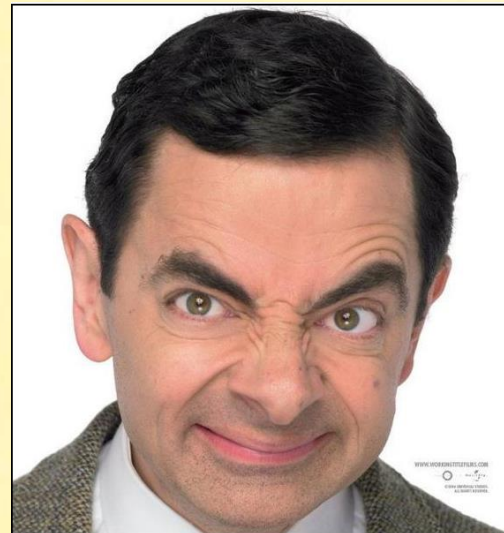
```
if (Robert / 4 - Charles / 3 == 11)
{
    cout << "\n Чарлз получил: " << Charles <<
        "\n Роберт получил: " << Robert;
    cout << endl;
}
return 0;
}
```

А вот и юридический отчёт о проделанной нами работе:



```
Математический фольклор. Задача Д39
Чарлз получил: 24
Роберт получил: 76
```

Английский юмор



Известно, что англичане хитрее американцев и всех остальных европейцев, поэтому и задачи у них с хитринками и подвохами.

В дом очаровательной мисс Чарити Локайер пришёл мистер Марк Девис, чтобы просить её руки. Его сопровождал товарищ. Мисс Локайер пригласила их в гостиную, подала 3 пустые чашки для чая и сахарницу, в которой было 10 кусочков сахара. Кавалеры помогли в сервировке стола. Мисс Локайер заявила, что отнесётся серьёзно к предложению мистера Девиса при одном условии: если он распределит 10 кусочков сахара по трём чашкам так, чтобы в каждой чашке было нечётное число кусков.

После некоторого смущения и краткого размышления мистер Девис нашёл верное решение. Какое?

Подобные задачи были известны в Англии ещё в XVII веке. Встречаются варианты со шляпами и корзинами.

Так как чашек было 3 – нечётное число, то чётное число кусочков сахара – 10 – нельзя между ними «распределить» без русской смекалки.



То есть после осахаривания чашек одну из них – с нечётным числом кусочков сахара – следует поставить в чашку с чётным числом кусочков сахара. Тогда в ней также будет нечётное число кусочков рафинада.

Обозначим чашку с чётным числом кусочков сахара буквой **a**, тогда остальные две чашки – с нечётным числом кусочков сахара – логично обозначить буквами **b** и **c**.

Из условия задачи следует, что число кусочков сахара в чашке **a** изменяется от 0 до 10, а в чашке **b** - от 1 до 10 – a. На чашку **c** приходится оставшиеся кусочки сахара:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д37");

    int var = 0;
    for (int a = 0; a <= 10; a += 2)
        for (int b = 1; b <= 10 - a; b += 2)
        {
            int c = 10 - a - b;
            cout << "Вариант #" << ++var << endl;
            cout << " a = " << a <<
                "\n b = " << b <<
                "\n c = " << c;
            cout << endl;
        }

    return 0;
}
```

Существует **15 способов** решения сахарной задачи коварной, но очаровательной мисс Локайер:

```
Математический фольклор. Задача Д37
Вариант #1
a = 0
b = 1
c = 9
Вариант #2
a = 0
b = 3
c = 7
Вариант #3
a = 0
b = 5
c = 5
Вариант #4
a = 0
b = 7
c = 3
Вариант #5
a = 0
b = 9
c = 1
Вариант #6
a = 2
b = 1
c = 7
Вариант #7
a = 2
b = 3
c = 5
```

```
Вариант #8
a = 2
b = 5
c = 3
Вариант #9
a = 2
b = 7
c = 1
Вариант #10
a = 4
b = 1
c = 5
Вариант #11
a = 4
b = 3
c = 3
Вариант #12
a = 4
b = 5
c = 1
Вариант #13
a = 6
b = 1
c = 3
Вариант #14
a = 6
b = 3
c = 1
Вариант #15
a = 8
b = 1
c = 1
```

Русские яблоки



А вот задачка про яблочки наливные!

Крестьянка несла на базар корзину яблок. Первому покупателю она продала половину всех яблок и ещё половину яблока, второму – половину от оставшихся яблок и ещё половину яблока, третьему – половину от нового остатка и ещё половину яблока и т.д. Когда шестой купил половину оставшихся яблок и ещё половину яблока, то яблок в корзине не осталось.

Сколько яблок было у крестьянки, если каждый купил целое число яблок?

У болгар есть аналогичная задача, но про **груши**.

Условие задачи настолько **циклично**, что мы просто обязаны применить цикл *for*, в котором 6 покупателей поочерёдно приобретают яблоки. Число яблок нам неизвестно, поэтому мы перебираем их во внешнем бесконечном цикле *for*, начиная с единицы. Как только крестьянка продаст всю корзину яблок, мы печатаем ответ и заканчиваем решение задачи:

```
#include <iostream>
#include<windows.h>

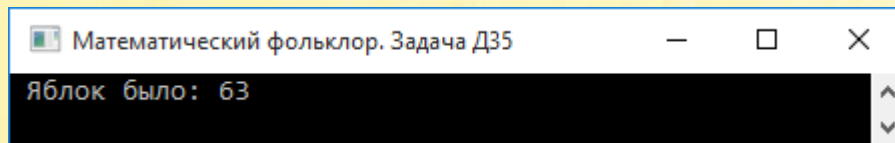
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д35");

    for (int apple = 1; ; ++apple)
    {
        double ost = apple;
        for (int i = 1; i <= 6; ++i)
        {
            ost = (ost - 1)/2;
        }
        if (ost == 0)
        {
            cout << " Яблок было: " << apple;
```

```
        cout << endl;
        break;
    }
}
return 0;
}
```

А ответ такой:



С задачей легко справиться вручную, если решать её **ретроспективно**, то есть начиная с конца.

Действительно, шестому покупателю досталась половина яблока и ещё столько же, то есть ровно 1 яблоко. После этого корзина опустела. Пятый купил в 2 раза больше + 1 яблоко, то есть 3. Четвёртый – $3*2 + 1 = 7$. И так далее – до победного начала.

Турецкие долгожители

Маленькое горное село славилось своими долгожителями. Особо уважали старого Исхана, имевшего детей, внуков, правнуков и праправнуков. Всего их вместе с Исханом было 2801. Праправнуки были ещё маленькие и не имели детей, а все остальные имели по одинаковому числу детей, и все дети были живы и здоровы.



Сколько детей имел старый Исхан?

Задача чисто программистская и легко решается простым перебором в бесконечном цикле *for*:

```
#include <iostream>
#include <windows.h>

using namespace std;

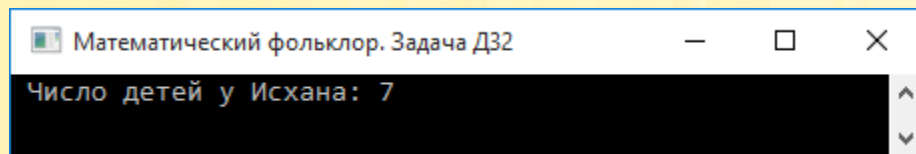
int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д32");

    for (int n = 1; ; ++n)
    {
        //n - число детей у Исхана
        //n * n - число внуков у Исхана
        //n * n * n - число правнуков у Исхана
        //n * n * n * n - число праправнуков у Исхана
        //всего - 2800 человек без Исхана:

        if (n + n * n + n * n * n + n * n * n * n == 2800)
        {
            cout << " Число детей у Исхана: " << n;
```

```
        cout << endl;  
        break;  
    }  
}  
  
return 0;  
}
```

Все детишки Исхана аккуратно пересчитаны:



Чешские сливы



Во дворец чешской королевы Любуши пришли трое просить руки её дочери. И она им предложила сначала ответить на такой вопрос:

- В корзине находятся сливы. Если первому из вас дать половину всех слив и ещё одну, второму – половину от оставшихся и ещё две сливы, третьему – половину от нового остатка и ещё три сливы, то слив в корзине не останется. Продолжать разговор я буду только с тем, кто ответит, сколько слив было в корзине.

Что должны ответить кандидаты?

Легко увидеть в этой задаче усложнённый вариант *Русских яблок*, поэтому в сжатые сроки мы переделываем исходный код и получаем достойный ответ:

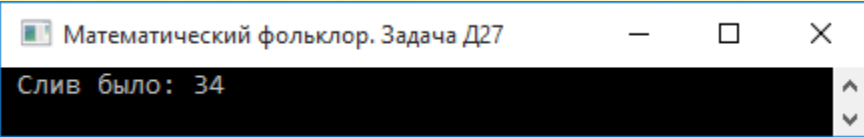
```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д27");

    for (int sliva = 1; ; ++sliva)
    {
        double ost = sliva;
        for (int i = 1; i <= 3; ++i)
        {
            ost = ost/2 - i;
        }
        if (ost == 0)
        {
            cout << " Слив было: " << sliva;
            cout << endl;
            break;
        }
    }

    return 0;
}
```



Математический фольклор. Задача Д27

Слив было: 34

Французский покупатель



Пьер был в хорошем настроении, имел некоторую сумму денег и решил пойти в магазин. У хозяина магазина он занял столько денег, сколько у него было, после чего потратил 1 франк. Затем он зашёл во второй магазин, снова занял у хозяина, сколько имел, и потратил 2 франка. Он посетил ещё два магазина, где занимал столько, сколько имел, и тратил соответственно 5 и 6 франков. Когда он вышел из последнего магазина, то оказалось, что денег у него нет.

Сколько денег было у Пьера первоначально и сколько он их всего потратил?

Если бы Пьер был осмотрительнее и не занимал деньги, то задача решалась бы точно так же, как с яблоками и сливами. Здесь же мы должны учесть приток капитала и нелинейные расходы Пьера в магазинах:

```
#include <iostream>
#include<windows.h>

using namespace std;
```



```

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д26");

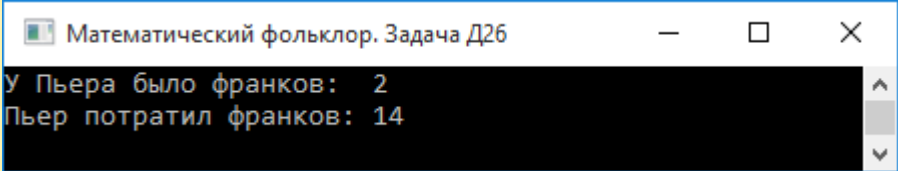
    int traty[] = { 0, 1, 2, 5, 6 };
    int money = 1;

    for (; ; ++money)
    {
        int sum = money;
        for (int n = 1; n <= 4; ++n)
        {
            sum = sum + sum - traty[n];
        }
        if (sum == 0)
        {
            cout << "У Пьера было франков: " << money << endl;
            cout << "Пьер потратил франков: " << 14;
            cout << endl;
            break;
        }
    }

    return 0;
}

```

Ответ на задачу весьма забавный: у Пьера было 2 франка, а потратил он 14!



```

Математический фольклор. Задача Д26
У Пьера было франков: 2
Пьер потратил франков: 14

```

Болгарский парикмахер



Трое мужчин пришли к парикмахеру. Побрив первого, парикмахер сказал:

- Посмотри, сколько денег в ящике стола и возьми 2 лева сдачи.

То же сказал парикмахер и второму, и третьему. После того как трое ушли, оказалось, что в кассе нет денег.

Сколько денег было в кассе перед тем, как заплатил первый мужчина?

В этой задаче расчёты лучше вести не в левах, а в стотинках, чтобы избежать неточных вычислений с данными типа *double*. А окончательный результат мы переведём в левы, как в книжном ответе:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача 110");

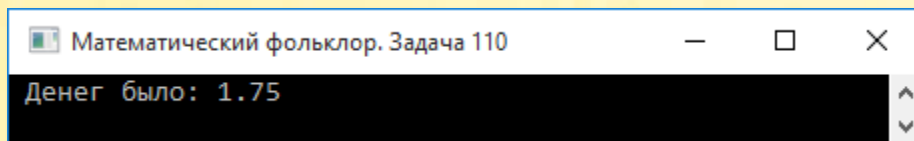
    for (int money = 100; ; ++money)
    {
        int ost = money;
```

```

for (int i = 1; i <= 3; ++i)
{
    ost = ost + ost - 200;
}
if (ost == 0)
{
    cout << " Денег было: " << money/100.0;
    cout << endl;
    break;
}
}
return 0;
}

```

Других сложностей задача нам не предлагает. А **ответ** такой: в кассе было 1,75 лева:



Болгарские сливы



Сливы – они и есть сливы: что чешские, что болгарские, поэтому мы легко справимся и с этой задачей!

Из корзины слив одна женщина взяла половину и ещё одну, другая женщина взяла половину от оставшихся и ещё одну, третья женщина взяла половину от последнего остатка и ещё 3 сливы, после чего в корзине слив не осталось.

Сколько слив было сначала в корзине?

Но нужно принять во внимание, что женщины брали нерегулярное число слив. Как обычно, для хранения слив и прочих фруктов мы используем **массив**:

```
#include <iostream>
#include<windows.h>

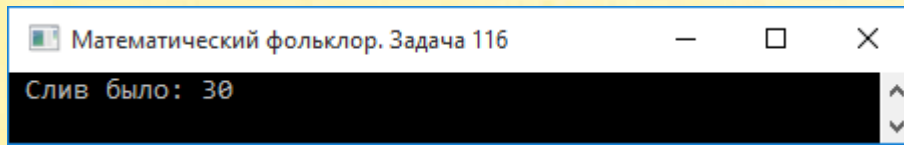
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача 116");

    int brali[] = { 0, 1, 1, 3 };
    for (int sliva = 1; ; ++sliva)
    {
        double ost = sliva;
        for (int i = 1; i <= 3; ++i)
        {
            ost = ost / 2 - brali[i];
        }
        if (ost == 0)
        {
            cout << " Слив было: " << sliva;
            cout << endl;
            break;
        }
    }

    return 0;
}
```

А было в корзине 30 слив:



Немецкий вопрос



Сын спросил отца, сколько ему лет. Отец ответил так:

- Если к моим годам прибавить их половину, затем их четверть и ещё один год, то получится 134 года.

Сколько лет отцу?

Нетрудно заметить, что возраст отца кратен 4. Перебираем все числа, делящиеся на 4 без остатка, в бесконечном цикле *for* до тех пор, пока не выполнится условие задачи:

```
#include <iostream>
#include <windows.h>

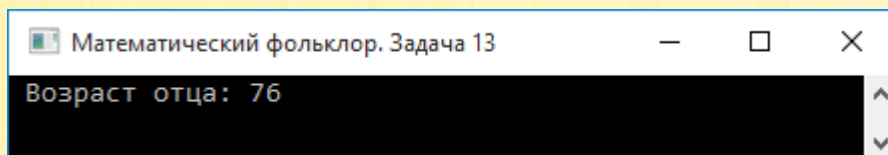
using namespace std;
```

```
int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача 13");

    for (int let = 0; ; let += 4)
    {
        if (let + let/2 + let/4 + 1 == 134)
        {
            cout << " Возраст отца: " << let << endl;
            cout << endl;
            break;
        }
    }

    return 0;
}
```

Наверняка у этого отца **взрослый** сын. Странно, что он не знает его возраста...



Индийские рупии



Один говорит другому:

- Дай мне 100 рупий, и я буду в 2 раза богаче тебя.

Второй отвечает:

- А если ты дашь мне 10 рупий, то я стану в 6 раз богаче тебя.

Сколько денег было у каждого?

Очевидно, что у **первого** было не меньше 10 рупий, а у **второго** – не меньше 100. Верхний предел нам неизвестен, но нельзя организовать 2 бесконечных вложенных цикла, поэтому во втором (но не в первом!) цикле мы полагаем, что у второго было не больше 1000 рупий. В случае неудачи мы повысим его денежное содержание.

По условию задачи, должны выполняться 2 условия, которые легко изложить на языке C++:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
```

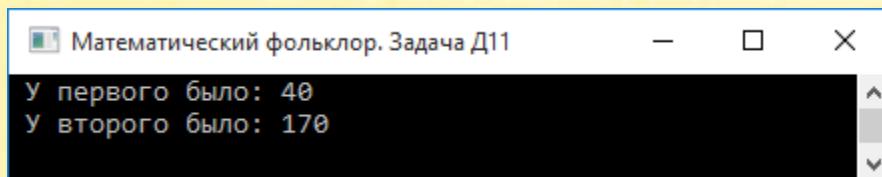
```

system("title Математический фольклор. Задача Д11");

for (int one = 10; ; ++one)
    for (int two = 100; two <= 1000; ++two)
    {
        bool uslovie1 = one + 100 == 2*(two-100);
        bool uslovie2 = two + 10 == 6*(one-10);
        if (uslovie1 && uslovie2)
        {
            cout << " У первого было: " << one << endl;
            cout << " У второго было: " << two << endl;
            cout << endl;
            return 0;
        }
    }
}

```

Итак, у первого было 40 рупий, а у второго – 170:



```

Математический фольклор. Задача Д11
У первого было: 40
У второго было: 170

```

Русские гуси



Очень популярная задача про стаю гусей:

Летела стая гусей, а навстречу им летит один гусь и говорит:

- Здравствуйте, сто гусей!

Старший гусь, их вожак, ответил:

- Нас не сто гусей. Но если взять сколько есть, да ещё столько, да ещё пол-столька, да четверть столька, да ещё вместе с тобой, нас будет 100.

Сколько было гусей?

Чтобы решить задачу, нужно просто внимательно прочитать её и записать условие на математическом языке:

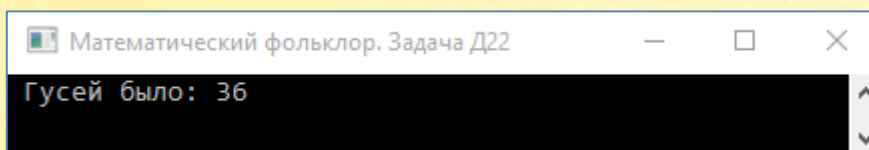
```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математический фольклор. Задача Д22");

    for (int gus = 1; ; ++gus)
    {
        if (gus + gus + gus/2 + gus/4 + 1 == 100)
        {
            cout << " Гусей было: " << gus;
            cout << endl;
            break;
        }
    }

    return 0;
}
```



У болгар имеется подобная задача про аистов (*Математический фольклор, Задача 124*):

Шёл человек с поля, а навстречу ему аисты.

- Здравствуйте, сто аистов, - поздоровался он.

А старший аист, их вожак, ответил:

- Нас не сто! А если к нам подлетит ещё столько, сколько нас, и ещё половина, и ещё четверть, и вместе с тобой нас станет 100!

Сколько было аистов?

В XVIII веке эта задача была популярна не только в России, но и в Германии, только место гусей в ней заняли голуби (*Математический фольклор, Задача Д21*):

Крестьянин шёл с поля, встретил стаю голубей и сказал:

- Добрый день, сто голубей.

Но один из голубей ответил:

- Нет, нас не сто! Но если взять ещё столько, ещё половину, ещё четверть, и вместе с тобой нас будет сто.

Сколько было голубей?

И неожиданная российская интерпретация этой же задачи (*Математический фольклор, Задача Д20*):

Отец спросил учителя своего сына, сколько у него учеников. Учитель ответил:

- Если взять ещё столько учеников, сколько уже есть, да ещё половину и четверть их, да ещё другого твоего сына, то станет точно 100.

Сколько у него учеников?

Насос Эдисона



Эта задача основана на историческом **анекдоте**. Известно, что калитки с насосом у Эдисона не было, да и быть не могло, поскольку её никто не смог бы открыть, и Эдисон быстро остался бы без друзей.

Томас Альва Эдисон обладал тонким чувством юмора. Его многочисленные посетители часто удивлялись, почему калитка в саду перед домом великого изобретателя открывается с трудом. Наконец, один из друзей спросил у Эдисона:

- Неужели такой технический гений, как ты, не может отрегулировать какую-то калитку?

- Калитка отрегулирована именно так, как надо, - смеясь возразил Эдисон. — Я сделал от неё привод к цистерне, и каждый, кто приходит ко мне, накачивает в цистерну 20 литров воды.

Если бы каждый посетитель вместо 20 литров накачивал в цистерну 25 литров воды, то для заполнения цистерны понадобилось бы на 12 человек меньше.

Сколько воды вмещает цистерна?

Задачу Эдисона решить гораздо легче, чем отворить его калитку. Достаточно переписать условие задачи на язык C++. При этом мы должны учесть, что в нём присутствует операция деления, поэтому мы выбираем **вещественный** тип для переменной бесконечного цикла *for*:

```
#include <iostream>
#include<windows.h>

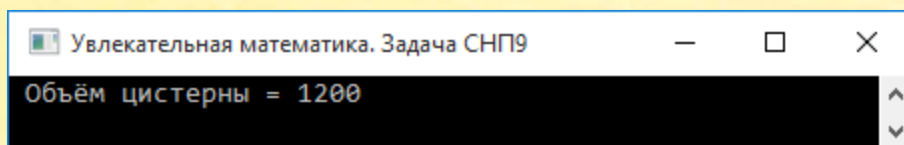
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика. Задача СНП9");

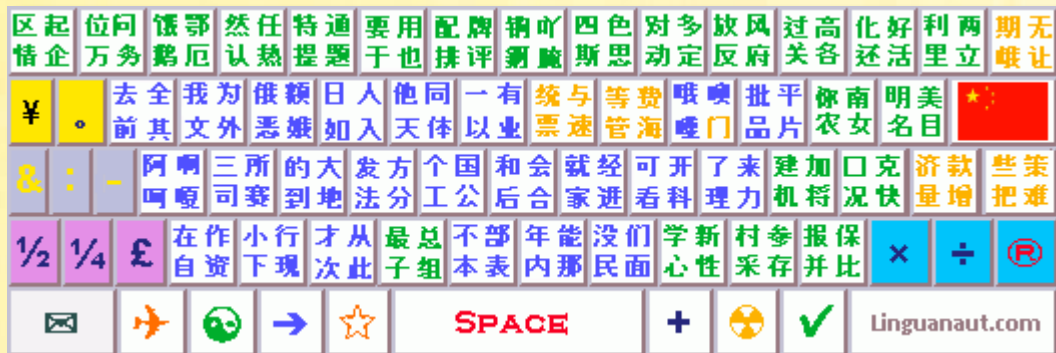
    for (double vol = 0.0; ; ++vol)
    {
        if (vol/25 == vol/20 - 12)
        {
            cout << " Объем цистерны = " << vol;
            cout << endl;
            break;
        }
    }

    return 0;
}
```

У большого мастера и цистерна была соответствующего объёма!



Китайская арифметика



Всем известно выражение *китайская грамота*, означающее сложные и непонятные вещи. Напротив, китайская арифметика проста, и понятна. Вот наглядный пример из трактата *Начала искусства вычисления*, написанного в 1593 году:

Найти число, которое при делении на 3 даёт в остатке 2, при делении на 5 даёт в остатке 3 и при делении на 7 – снова 2.

Мы перебираем целые неотрицательные числа в бесконечном цикле *for*, пока не найдём требуемое число:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Наука и жизнь 1963-03-38-3");

    int var = 0;
    for (int num = 0; ; ++num)
    {
        if (num % 3 == 2 && num % 5 == 3 && num % 7 == 2)
        {
            cout << " Число = " << num << endl;
        }
    }
}
```

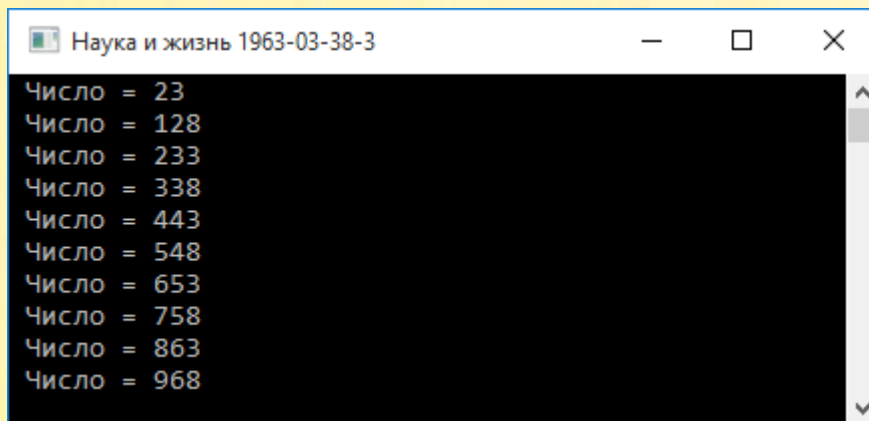
```
        if (++var == 10)
            break;
    }
}
cout << endl;

return 0;
}
```

Минимальное число равно 23, а остальные можно найти по формуле:

Китайское число = $23 + 105n$,

где n – целое неотрицательное число.



```
Наука и жизнь 1963-03-38-3
Число = 23
Число = 128
Число = 233
Число = 338
Число = 443
Число = 548
Число = 653
Число = 758
Число = 863
Число = 968
```

Задача Этьена Безу

Этьен Безу – французский математик 18 века - преподавал математику в Училище гардемаринов и в Королевском артиллерийском корпусе.

Нам известна такая задача Этьена Безу:

По контракту работникам причитается по 48 франков за каждый отработанный день, а за каждый неотработанный день с них взывается по 12 франков. Через 30 дней выяснилось, что работникам ничего не причитается.

Сколько дней они отработали в течение этих 30 дней?



Пусть они отработали x дней.

За эти дни им причитается по $48x$ франков.

Остальные $(30 - x)$ дней были «выходными».

За них работники должны вернуть $12(30 - x)$ франков.

Всего за 30 дней каждому работнику причитается 0 франков:

$$48x - 12(30 - x) = 0$$

Так как число дней – **целое**, то уравнение решается простым перебором в бесконечном цикле **for**. Когда условие задачи выполнится, цикл прекращается.

Вместо математической переменной x мы создадим программистскую переменную **days**. Вот и вся задача:

```
#include <iostream>
#include<windows.h>
```

```

using namespace std;

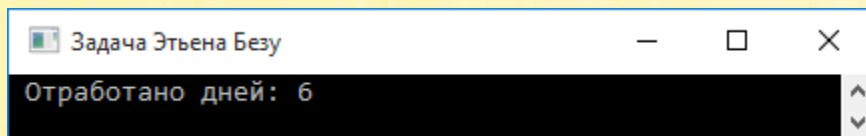
int main()
{
    SetConsoleOutputCP(1251);
    system("title Задача Этьена Безу");

    for (int days = 0; ; ++days)
    {
        //cout << 48 * days << endl;
        //cout << 12 * (30 - days) << endl;
        if (48 * days - 12 * (30 - days) == 0)
        {
            cout << " Отработано дней: " << days << endl;
            break;
        }
    }
    cout << endl;

    return 0;
}

```

А ответ такой:



Задачу легко решить в уме.

Работники за 1 отработанный день получают в 4 раза больше, чем с них взыскивают за каждый неотработанный день. В итоге они не заработали ничего. Это значит, что они отработали в 4 раза меньше дней, чем «прогуляли». Это составляет одну пятую от 30 дней, то есть 6 дней.

Кому сколько лет?

Задача 70 из книги *Русские головоломки*:

Дед, отец и сын во время прогулки встретили знакомого, который спросил их, сколько им лет.

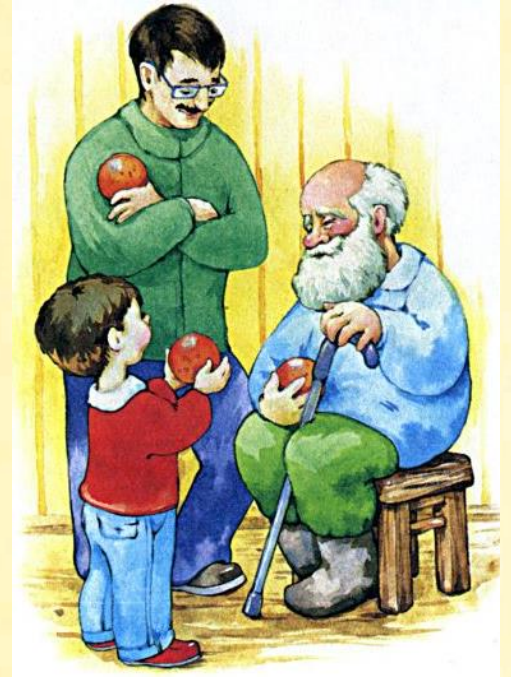
- Нам 121 год, - ответил за всех дед и важно зашагал вперёд.

Тогда знакомый, продолжая интересоваться их возрастом, спросил отца:

- Ну, скажите же, сколько вам лет?

- Мне вместе с сыном 44 года, - отвечал отец, - а сын на 28 лет моложе меня.

Так знакомому и не пришлось узнать, сколько лет каждому из них. Не сообразите ли вы?



Легко сообразить, что **деду** $121 - 44 = 77$ лет, потому что остальные 44 года из общей суммы приходятся на отца и сына.

Пусть отцу x лет, а сыну – y . Вместе им 44 года:

$$x + y = 44$$

Условие, что сын на 28 лет моложе отца, даёт нам второе уравнение:

$$y = x - 28$$

Заменив математические иксы-игреки более осмысленными переменными **otets** и **syn**, мы быстро составляем программу:

```
#include <iostream>
#include<windows.h>

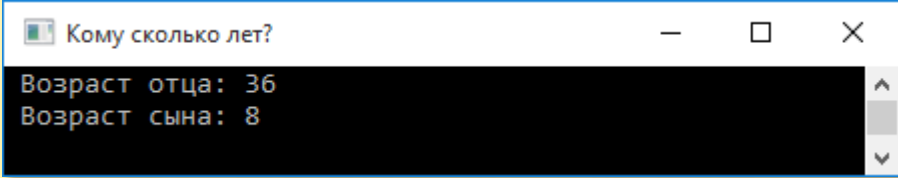
using namespace std;
```

```
int main()
{
    SetConsoleOutputCP(1251);
    system("title Кому сколько лет?");

    for (int syn = 1; ; ++syn)
    {
        int otets = 28 + syn;
        if (otets + syn == 44)
        {
            cout << " Возраст отца: " << otets << endl;
            cout << " Возраст сына: " << syn << endl;
            break;
        }
    }
    cout << endl;

    return 0;
}
```

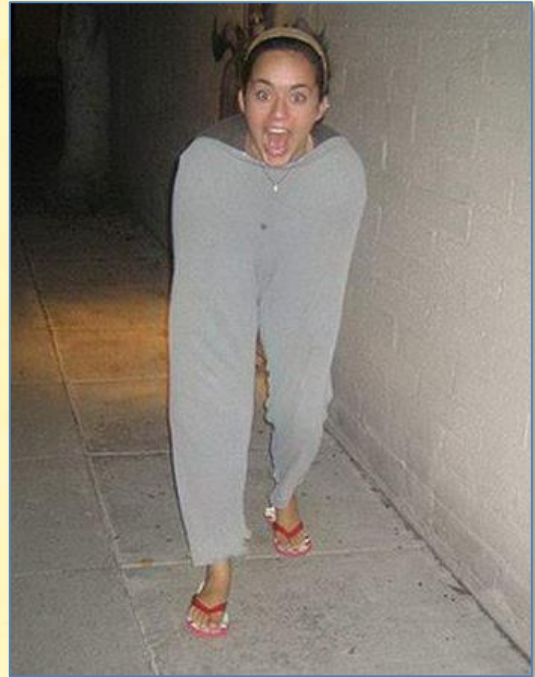
Ответ не заставил себя долго ждать:



```
Кому сколько лет?
Возраст отца: 36
Возраст сына: 8
```

Как-то даже неудобно перед компьютером, что мы задаём ему такие простые задачи...

Ноги и головы



Задача 71 из книги *Русские головоломки*:

На скотном дворе гуляли гуси и поросята. Хозяин и его сын вышли на двор, посмотрели на живность и пошли в поле. По дороге сын и спрашивает:

- Отец, сколько у нас на скотном дворе гусей и сколько поросят?

- А вот угадай-ка сам, - ответил отец. – Если считать по головам, то на дворе 25 голов, а если по ногам, то 70 ног.

Сколько было гусей и сколько поросят?

Эта задача очень похожа на кроличье-фазанью, которых здесь заменили поросята и гуси:

```
#include <iostream>
#include <windows.h>

using namespace std;
```

```

int main()
{
    SetConsoleOutputCP(1251);
    system("title Русские головоломки. Задача 71");

    //общее число голов:
    const int HEADS = 25;
    //общее число ног:
    const int LEGS = 70;

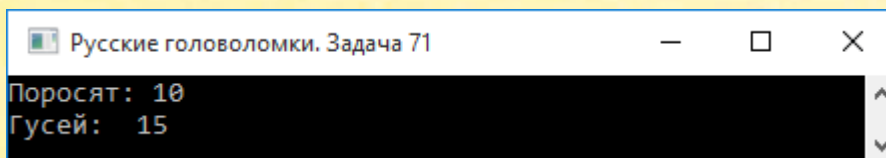
    //макс. число поросят:
    int maxPoros = LEGS / 4;

    //решаем задачу:
    for (int i = 0; i <= maxPoros; ++i)
    {
        //число гусей:
        int nGus = HEADS - i;
        if (i * 4 + nGus * 2 == LEGS)
        {
            cout << "Поросят: " << i << endl;
            cout << "Гусей:  " << nGus;
        }
    }
    cout << endl;

    return 0;
}

```

Задача простейшая даже для компьютера. Инвентаризация закончена мгновенно: **гусей было 15 голов, а поросят – 10:**



```

Русские головоломки. Задача 71
Поросят: 10
Гусей: 15

```

Задания для самостоятельного решения

Задача Фольклор Д43

Бразильский кофе

Каждое утро Рейнальдо выпивает 1 чашку крепкого кофе, которая содержит определённую дозу кофеина. Однажды ему предстояло приготовить себе традиционный кофе из партии зёрен, из которых уже было извлечено 97% кофеина.

Сколько чашек такого кофе должен выпить Рейнальдо, чтобы в них содержалось столько кофеина, сколько содержалось его в одной чашке крепкого кофе?

Простая задача на пропорции.

Ответ: 33 $\frac{1}{3}$ чашки.

Задача Фольклор Д29

Возраст сыновей

Когда у отца спросили, сколько лет его сыновьям, он ответил:

- Старшему в три раза больше, чем младшему, а им вместе столько лет, сколько было мне 29 лет назад. Сейчас мне 45 лет.

Сколько лет сыновьям?

29 лет назад отцу было 16 лет. Отсюда следует...

Ответ: 12 лет и 4 года.

Задача Фольклор Д25

Который час?

На вопрос: «Сколько времени?» - был дан такой ответ:

- Две пятых времени, прошедшего от полночи до этого момента, равно двум третьим времени, которое осталось до полудня.

Сколько сейчас времени?

От полночи до полудня 12 часов. Если от полночи прошло x часов, то до полудня осталось $(12-x)$ часов. Осталось составить простейшее уравнение с одним неизвестным.

Ответ: 7 часов 30 минут.

Задача Фольклор Д24

Австралийский скотовод

Скотовод завещал трём своим сыновьям – Альфреду, Джону и Чарльзу – разделить стадо овец следующим образом: Альфред получит на 20% больше Джона и на 25% больше Чарльза. Часть Джона – 3600 овец.

Сколько овец получит Чарльз?

Простая задача, решение которой не требует дополнительных пояснений.

Ответ: 3456.

Задача Фольклор Д23

Французские братья

Три брата хотели купить дом, который стоил 26000 франков. Условились, что первый даст половину, второй – треть, а третий – четверть стоимости.

Сколько денег даст каждый?

Легко посчитать, что $\frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{13}{12}$, что соответствует 26000 франков. Теперь найти долю каждого брата не составит труда.

Ответ: 12000, 8000 и 6000.





СОВРЕМЕННЫЕ ЗАДАЧИ

Современных занимательных задач и не сосчитаешь! Их можно найти и в книгах, и в журналах, и в газетах, и в Интернете. Фантазии авторов нет предела. Нет ни одного раздела математики, который не попал бы в поле зрения составителей головоломных задач.

В этой главе вы снова встретитесь с задачами из разных стран мира. Решайте их - и от математики можно получать удовольствие. Да ещё какое!

Кузнечики

В седьмом номере журнала *Квантик* за этот год, на странице 32 напечатана вот такая конкурсная задача:



31. Двадцать пять ребят пошли в лес и стали ловить кузнечиков. Несколько ребят поймали по одному кузнечику, половина оставшихся ребят поймали по два кузнечика, а остальные не смогли поймать ни одного. Сколько всего кузнечиков поймали ребята?

Задача скорее на сообразительность, чем математическая. Такие задачи очень простые, да вот только догадаться сложно, как же их решать.

Можно решить задачу **алгебраически**, обозначив буквой x число ребят, которые поймали по одному кузнечику. Но с *C++* задачу лучше решать простым **перебором**.

Мы знаем, что x не меньше 1 и не больше 25. Причём это число **нечётное**, иначе число оставшихся ребят также будет нечётным, а от такого числа нельзя взять **ровно половину**.

Создаём переменную x . Её начальное значение равно 1. В цикле **for** мы изменяем значение переменной x на 2, чтобы оно всегда было нечётным:

```
#include <iostream>
#include<windows.h>

using namespace std;
```

```
int main()
{
    SetConsoleOutputCP(1251);
    system("title Кузнечики");

    for (int x = 1; x <= 25; x += 2)
    {
```

Создайте переменную **nOstReb**.

Число оставшихся ребят равно: $25 - x$:

```
int nOstReb = 25 - x;
```

В цикле *for* переменная **nOstReb** принимает все возможные значения:

```
for (int x = 1; x <= 25; x += 2)
{
    int nOstReb = 25 - x;
}
```

Создайте переменную **nKuzn** для подсчёта пойманных кузнечиков.

По условию задачи, число кузнечиков равно: $x + (25 - x) / 2 * 2$:

```
double nKuzn = x + (25 - x) / 2 * 2;
```

Из уравнения число кузнечиков можно найти в уме. Но мы не будем торопить события и запишем это выражение в цикле *for*:

```
for (int x = 1; x <= 25; x += 2)
{
    int nOstReb = 25 - x;
    double nKuzn = x + (25 - x) / 2 * 2;
}
```

Результаты вычислений мы напечатаем на экране:

```

#include <iostream>
#include<windows.h>

using namespace std;

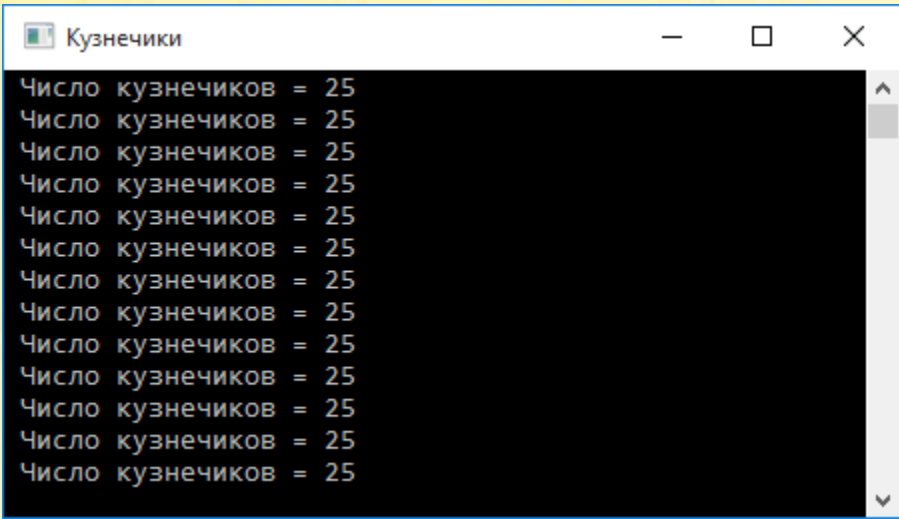
int main()
{
    SetConsoleOutputCP(1251);
    system("title Кузнечики");

    for (int x = 1; x <= 25; x += 2)
    {
        int nOstReb = 25 -x;
        double nKuzn = x + (25 - x) / 2 * 2;
        cout << " Число кузнечиков = " << nKuzn << endl;
    }
    cout << endl;

    return 0;
}

```

Запускаем программу и смотрим на список:



```

Кузнечики
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25
Число кузнечиков = 25

```

Оказывается, каким бы ни было число ребят, поймавших по 1 кузнечику, **общее число кузнечиков всегда равняется 25.**

А теперь легко догадаться, почему так происходит. Выражение

$$x + (25 - x) / 2 * 2$$

при любом значении x равно 25:

$$x + (25 - x) = 25$$

Задача пустяковая, а ведь не сразу это поймёшь!

Бельгийские числа



Типичная **переборная задача** на поиск чисел, цифры которых удовлетворяют некоторому условию:

Найти трёхзначные числа вида abc , цифры которых удовлетворяют уравнению:

$$a^2 - b^2 - c^2 = a - b - c$$

Все 3 цифры числа должны быть различны.

Как обычно в таких случаях, выписываем 3 вложенных цикла, в которых и перебираем все возможные комбинации цифр-

Так как числа **трёхзначные**, то первая цифра не может быть нулём:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика. Задача 12");

    for (int d1 = 1; d1 < 10; ++d1)
```

А вторая и третья должны отличаться друг от друга и от первой цифры:

```
for (int d2 = 0; d2 < 10; ++d2)
{
    if (d2 == d1)
        continue;

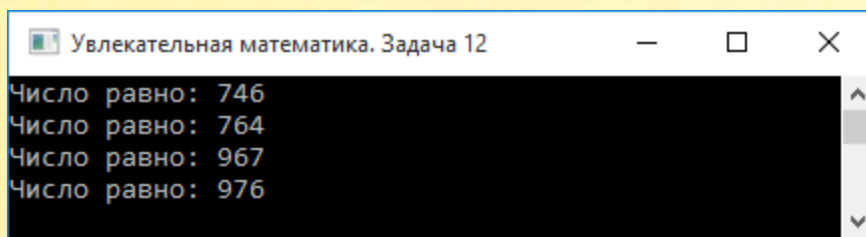
    for (int d3 = 0; d3 < 10; ++d3)
    {
        if (d3 == d2 || d3 == d1)
            continue;
```

Когда условие задачи выполнится, мы напечатаем очередное решение:

```
        if (d1 * d1 - d2 * d2 - d3 * d3 == d1 - d2 - d3)
        {
            cout << "Число равно: " <<
                (d1 * 100 + d2 * 10 + d3) << endl;
        }
```

```
}  
    }  
    }  
    return 0;  
}
```

На рисунке вы видите все **четыре решения** этой бесхитростной задачи:



```
Увлекательная математика. Задача 12  
Число равно: 746  
Число равно: 764  
Число равно: 967  
Число равно: 976
```

Немецкая копилка



Как говорится, опять за рыбу деньги!

Отец обещал сыну за каждую правильно решённую задачу опускать в копилку по 10 пфеннигов. За каждую неправильно решённую задачу сын должен возвращать отцу по 5 пфеннигов. После того как было решено 20 задач, у сына в копилке оказалось 80 пфеннигов.

Сколько задач сын решил неправильно и сколько без единой ошибки?

Задача очень простая, поэтому не будем тратить компьютерное и собственное время на излишние комментарии:

```
#include <iostream>
#include<windows.h>

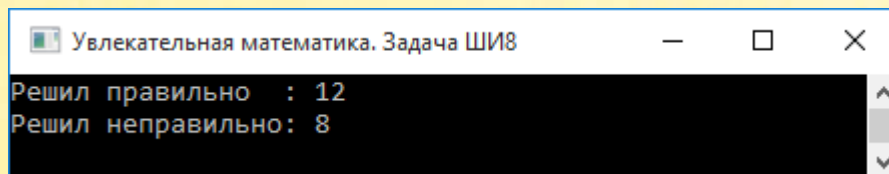
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Увлекательная математика. Задача ШИ8");

    for (int verno = 0; verno <= 20; ++verno)
    {
        if (verno * 10 - (20 - verno) * 5 == 80)
        {
            cout << "Решил правильно : " << verno << endl;
            cout << "Решил неправильно: " << (20 - verno);
            cout << endl;
        }
    }

    return 0;
}
```

Конечно, 80 пфеннигов деньги небольшие, но, судя по ответу, сынок и на них не наработал!



```
Увлекательная математика. Задача ШИ8
Решил правильно : 12
Решил неправильно: 8
```

Ошибки



Микрокалькулятор с ЕГГОГом

И ещё одна задачка из школьной жизни:

Марина сделала в диктанте несколько ошибок. Гриша у неё всё списал да ещё допустил 5 ошибок.

Сколько ошибок допустил каждый, если учитель обнаружил в двух диктантах 35 ошибок?

Объявим переменную `error` для подсчёта числа ошибок и инициализируем её нулём:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Ошибки");

    //число ошибок:
```



```
int error = 0;
```

В старых микрокалькуляторах английское слово **ERROR** (*ошибка*) на экране выглядело, как русское, но совершенно непонятное слово **ЕГГОГ**.

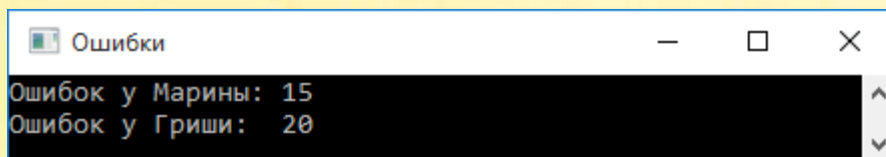
Будем добавлять по одной ошибке в цикле *do-while* до тех пор, пока не выполнится условие задачи, то есть число ошибок у Марины и у Гриши вместе взятых не достигнет 35:

```
do
{
    ++error;
}
while (error + (error + 5) != 35);
```

Когда это событие произойдёт, мы напечатаем ответ на экране:

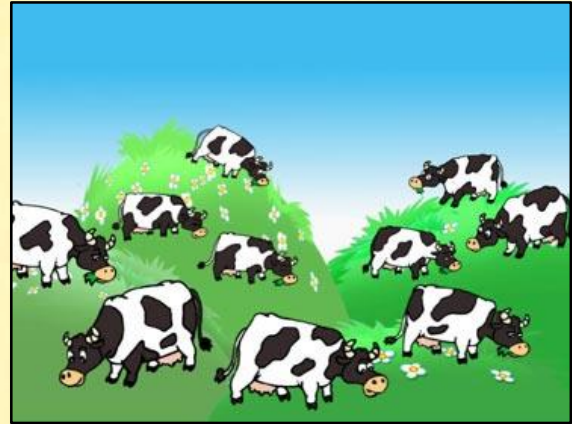
```
cout << "Ошибок у Марины: " << error << endl;
cout << "Ошибок у Гриши:  " << (error + 5) << endl;

return 0;
}
```



```
Ошибки
Ошибок у Марины: 15
Ошибок у Гриши: 20
```

Коровы



А теперь решим задачку из сельской жизни:

На лугу паслось несколько коров. У них ног на 24 больше, чем голов.

Сколько коров паслось на лугу?

Ответ на задачу мы найдём в условии задачи: коров паслось несколько. Это, конечно, шутка, а задачка-то ведь нешуточная!

Обозначим число коров через k :

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Коровы");

    //число коров:
    int k = 0;
```

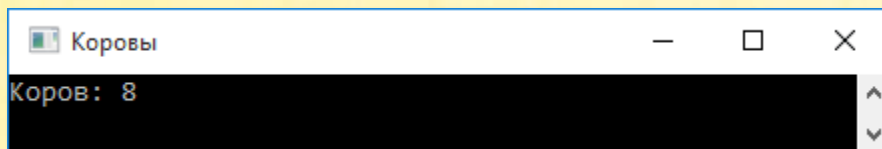
Будем добавлять по одной корове до тех пор, пока не выполнится условие задачи, то есть число ног у коров за вычетом числа самих коров не станет равно 24:

```
do
{
    ++k;
}
while (k * 4 - k != 24);

cout << "Коров: " << k << endl;

return 0;
}
```

Ответ: на лугу паслось 8 коров:.



Мешки с сокровищами



А вот задача из книги Катта Мурти (Katta G. Murty) *Junior Level Web-Book Optimization Models for decision Making*, Chapter 7:

7.8.2: A merchant has bags of emeralds (nine to a bag), and rubies (four to a bag). He has a total of 59 jewels. Required to find how many of his jewels are rubies.

Её можно перевести так:

У торговца есть сумки: с изумрудами (по 9 штук в сумке) и с рубинами (по 4 штуки в сумке). В общей сложности у него 59 драгоценностей.

Требуется найти, сколько из них - рубины.

Для общего числа драгоценностей мы введём константу **TOTAL**:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Мешки с сокровищами");

    //общее число драгоценностей:
    const int TOTAL = 59;
```

Число сумок с драгоценностями может изменяться от нуля до какого-то предельного значения, которое легко вычислить простым делением общего числа драгоценностей *TOTAL* на число драгоценностей в каждой сумке. Как вы помните, это 9 для **изумрудов** (*Emeralds*) и 4 – для **рубинов** (*Rubies*):

```
//макс. число сумок с изумрудами:
int maxEmeralds = TOTAL / 9;
//макс. число сумок с рубинами:
int maxRubies = TOTAL / 4;
```

С тем же успехом для этих целей можно использовать не переменные, а **константы**.

Теперь мы можем во вложенных циклах *for* перебрать все варианты числа сумок:

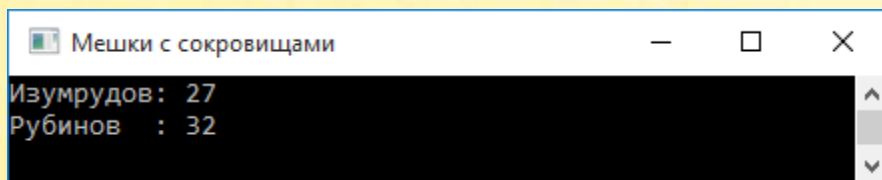
- с изумрудами – переменная цикла *j* - от 0 до *maxEmeralds*
- с рубинами – переменная цикла *i* - от 0 до *maxRubies*

Поскольку переменные циклов обозначают число мешков, то мы легко определим, сколько в них было драгоценностей. Для этого число сумок с изумрудами нужно умножить на 9, а число сумок с рубинами – на 4. Если в сумме они дадут 59 (*TOTAL*), то задача решена, и можно печатать ответ:

```
//решаем задачу:
for (int j = 0; j <= maxEmeralds; ++j)
    for (int i = 0; i <= maxRubies; ++i)
        if (j * 9 + i * 4 == TOTAL)
            cout << "Изумрудов: " << (j * 9) <<
                "\nРубинов : " << (i * 4) << endl;

return 0;
}
```

А ответ такой: рубинов у торговца было 32 штуки.



```
Мешки с сокровищами
Изумрудов: 27
Рубинов : 32
```

Повторяющиеся цифры



Росс Хонсбергер в книге **Математические изюминки [ХР92]** на страницах 126-127 предлагает решить такую задачу:

Определите количество цифр в длиннейшей последовательности ненулевых одинаковых цифр, которой может оканчиваться полный квадрат, **и найдите наименьший квадрат**, который оканчивается на такую максимальную последовательность.

Элементарная проверка показывает, что квадраты натуральных чисел могут заканчиваться только цифрами **0, 1, 4, 5, 6** и **9**. Из этого набора мы сразу должны исключить **ноль**.

Дальнейшие рассуждения смотрите в книге. Из них следует, что квадрат не может заканчиваться двумя единицами, пятёрками, шестёрками и девятками. Не может он заканчиваться и четырьмя четвёрками. Следовательно, максимальная последовательность равна **...444**.

Мы не можем и не должны сомневаться в строгом доказательстве Росса Хонсбергера, поэтому просто напишем программу, которая умеет находить квадраты, оканчивающиеся тремя одинаковыми цифрами (мы не ограничиваем себя только четвёрками).

В главной функции мы без труда выделяем из квадрата одну и три его последние цифры. Если все они одинаковы, то частное от деления трёх последних цифр на самую последнюю должно равняться 111. Дополнительно мы проверяем, чтобы последняя цифра не оказалась нулём:

```

#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Повторяющиеся цифры");

    int nVar = 0;
    for (long i = 10; i < 10000 /*1000000*/; ++i)
    {
        //последние 3 цифры:
        long n3 = i * i % 1000;
        //последняя цифра:
        long last = i * i % 10;
        if (last != 0 && n3 / last == 111)
        {
            ++nVar;
            cout << i << " в квадрате = " << i * i << endl;
        }
    }
    cout << "Всего найдено чисел: " << nVar;
    cout << endl;

    return 0;
}

```

Из миллиона первых квадратов только 4000 заканчиваются на 3 четвёрки. Вот небольшая часть списка:

```
Повторяющиеся цифры
38 в квадрате = 1444
462 в квадрате = 213444
538 в квадрате = 289444
962 в квадрате = 925444
1038 в квадрате = 1077444
1462 в квадрате = 2137444
1538 в квадрате = 2365444
1962 в квадрате = 3849444
2038 в квадрате = 4153444
2462 в квадрате = 6061444
2538 в квадрате = 6441444
2962 в квадрате = 8773444
3038 в квадрате = 9229444
3462 в квадрате = 11985444
3538 в квадрате = 12517444
3962 в квадрате = 15697444
4038 в квадрате = 16305444
4462 в квадрате = 19909444
4538 в квадрате = 20593444
4962 в квадрате = 24621444
5038 в квадрате = 25381444
5462 в квадрате = 29833444
5538 в квадрате = 30669444
5962 в квадрате = 35545444
6038 в квадрате = 36457444
6462 в квадрате = 41757444
6538 в квадрате = 42745444
6962 в квадрате = 48469444
7038 в квадрате = 49533444
7462 в квадрате = 55681444
7538 в квадрате = 56821444
7962 в квадрате = 63393444
8038 в квадрате = 64609444
8462 в квадрате = 71605444
8538 в квадрате = 72897444
8962 в квадрате = 80317444
9038 в квадрате = 81685444
```

Квадраты 1,4,9

В книге Айлена Варди (Ilan Vardi) *Computational Recreations in Mathematica* предлагается такое упражнение (*Exercise 2.2*, страница 20) для самостоятельного решения:

Найдите полные квадраты натуральных чисел, которые состоят только из цифр 1, 4 и 9.

В книге, на страницах 234-237 вы найдёте довольно сложное решение этой задачи для чисел до 10^{42} включительно. Мы упростим себе задачу, ограничив поиски только числами типа *unsigned long long*. Впрочем, мы не найдём только 2 числа:

```
444411911999914911441
419994999149149944149149944191494441
```

В главной функции программы мы перебираем все числа от 1 до максимально возможного для выбранного нами типа. Квадрат каждого числа мы проверяем в функции *Test* на предмет наличия в нём посторонних примесей – цифр, отличных от 1, 4 и 9:

```
#include <iostream>
#include<windows.h>
#include<math.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Квадраты 1-4-9");

    int nVar = 0;
    unsigned long long lmax = pow(2,64);
    for (unsigned long long i = 1; i <= lmax; ++i)
    {
        if (Test(i * i))
        {
            ++nVar;
            cout << i << " в квадрате = " << i * i << endl;
            if (nVar == 15)
                break;
        }
    }

    cout << "Всего найдено чисел: " << nVar;
```

```
    cout << endl;

    return 0;
}
```

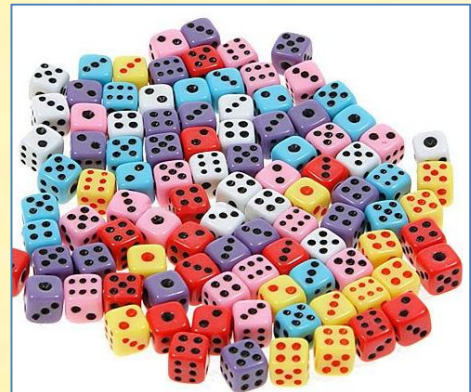
Проверка очень простая. Мы «отщипываем» от заданного числа его цифры, начиная с хвоста. Как только нам в руки попадётся негодная цифра, мы прекращаем поиски с отрицательным результатом. И только если число целиком и полностью состоит дозволённых цифр, функция возвращает *true*:

```
//ПРОВЕРЯЕМ ЦИФРЫ КВАДРАТА
bool Test(unsigned long long num)
{
    bool res = true;
    while (num > 0)
    {
        unsigned long long dig = num % 10;
        if (dig != 1 && dig != 4 && dig != 9)
        {
            res = false;
            break;
        }
        num /= 10;
    }
    return res;
}
```

Поиски чисел заняли некоторое время (весьма небольшое для такого рода задач), которое всё же не было потрачено напрасно – мы нашли 14 (тоже хорошее число, но не квадрат!) квадратов, состоящих из цифр 1, 4 и 9:

```
Квадраты 1-4-9
1 в квадрате = 1
2 в квадрате = 4
3 в квадрате = 9
7 в квадрате = 49
12 в квадрате = 144
21 в квадрате = 441
38 в квадрате = 1444
107 в квадрате = 11449
212 в квадрате = 44944
31488 в квадрате = 991494144
70107 в квадрате = 4914991449
387288 в квадрате = 149991994944
95610729 в квадрате = 9141411499911441
446653271 в квадрате = 199499144494999441
3148717107 в квадрате = 9914419419914449449
```

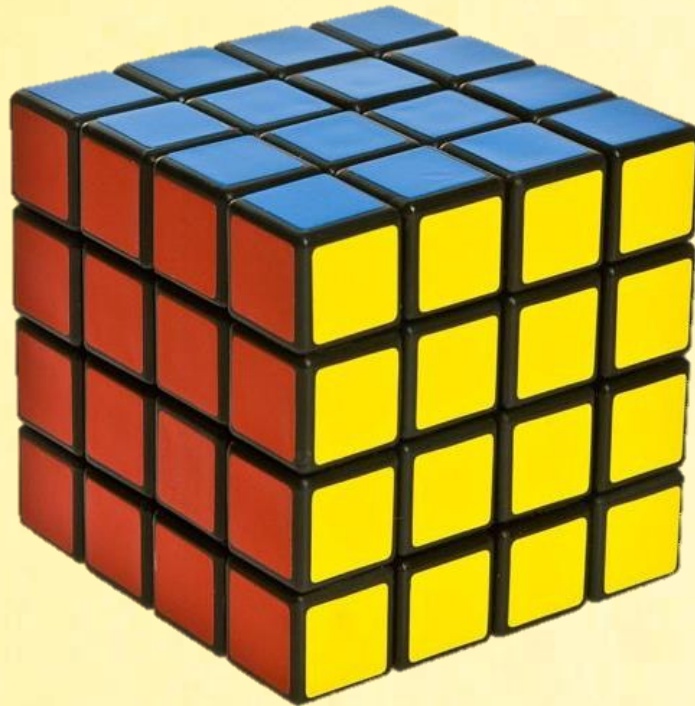
Кубики из кубиков



В журнале *Parade Magazine* от 15 декабря 2013 года предлагается решить такую задачу на пространственное воображение:

Consider a cubical structure composed of unit cubes (like a child's building blocks), four on an edge. Without drawing the structure or actually assembling it, how many cubical shapes can you envision within it?

Иначе говоря, нам предстоит мысленно **пересчитать все возможные кубы** в большом кубе, состоящем из $4 \times 4 \times 4$ маленьких кубиков:



Возьмём самый маленький кубик $1 \times 1 \times 1$, который находится в левом верхнем углу большого куба. Мы можем передвигать его по осям X , Y и Z от позиции 0 до позиции 4.

Куб $2 \times 2 \times 2$ мы можем перемещать аналогично, но уже только до позиции 3, иначе он выйдет за границы большого куба.

Кубы $3 \times 3 \times 3$ и $4 \times 4 \times 4$ допускают точно такие же действия - с соответствующими ограничениями.

Пусть большой куб имеет размеры сторон в элементарных кубиках **size**, тогда в функции *Solve* мы подсчитываем число кубов заданного размера $1..size$ и суммируем в переменной **answer**:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
```

```

{
    SetConsoleOutputCP(1251);
    system("title Кубики из кубиков");

    int size = 4;
    int answer = 0;
    //решаем задачу:
    for (int n = 1; n <= size; ++n)
    {
        int ncube = Solve(size, n);
        answer += ncube;
        cout << "Число кубиков " << n << "x" << n << "x" <<
            n << " равно " << ncube << endl;
    }
    cout << endl;
    cout << "Общее число кубиков равно " << answer;
    cout << endl;

    return 0;
}

```

Подсчёт кубов заданного размера `sizeSmall` в кубе `sizeBig` мы проводим в трёх вложенных циклах *for*:

```

//РЕШАЕМ ЗАДАЧУ
int Solve(int sizeBig, int sizeSmall)
{
    int res = 0;
    for (int x = 0; x <= sizeBig - sizeSmall; ++x)
        for (int y = 0; y <= sizeBig - sizeSmall; ++y)
            for (int z = 0; z <= sizeBig - sizeSmall; ++z)
                {
                    ++res;
                }
    return res;
}

```

Всего в кубе 4 x 4 x 4 мы насчитали ровно **100** кубов разных размеров:

```
Кубики из кубиков
Число кубиков 1x1x1 равно 64
Число кубиков 2x2x2 равно 27
Число кубиков 3x3x3 равно 8
Число кубиков 4x4x4 равно 1
Общее число кубиков равно 100
```

Глядя на этот ряд чисел, легко догадаться, что в кубе 5 x 5 x 5 на 125 кубов больше. И так далее...

Бизнес на мышах

В журнале *New Scientist* #1224 от 23 октября 1980 года была напечатана задача *Enigma 81. Blind mice*:



Johnny and Willie have hit on a simple scheme to get rich quick. Each has agreed to invest his pocket money in the purchase of a white mouse. Then they will combine their investments and sit back and wait for the multiplier to take effect. A spot of asset stripping will then set them up to plunge into guinea pigs at even greater profit.

The venture is somewhat riskier than they realise. Johnny will be buying his mouse at the Ace Pet Shop, where there is at present a stock of eight white mice, six of which are males and two females. Willie will be using the Peerless Pet Store, where the stock of white mice also numbers eight. Economics being more in their line than biology, each will be buying a mouse at random. But I dare say that all will be well, since it is 11 to 5 on that the mice will be of different sex.

How many females are in stock at the Peerless Pet Store?

На русский язык смысл её можно перевести так:

Джонни и Вилли решили быстро разбогатеть и для этого купили по одной белой мышке (а это оказались морские свинки). Джонни купил мышку в одном магазине, а Вилли в другом. В каждом магазине было по 8 белых мышей. В первом магазине было 6 самцов и 2 самки.

Сколько было самок во втором магазине, если Джонни и Вилли выбирали мышку наугад, а вероятность того, что купленные мышки имеют разный пол, равна 11 : 5?

Вероятность того, что **Джонни** купит **самца**, равна:

$$M1 = 6/8,$$

а **самку**:

$$F1 = 2/8$$

Если мы обозначим через **f** число мышек-самок во втором магазине, то вероятность того, что **Вилли** купит самку, равна:

$$F2 = f/8$$

а **самца**:

$$M2 = (8-f)/8$$

Вероятность покупки двух самцов равняется произведению вероятностей:

$$M1 \times M2$$

А вероятность покупки двух самок – произведению вероятностей:

$$F1 \times F2$$

Общая вероятность покупки однополых мышек равна сумме этих вероятностей:

$$p1 = M1 \times M2 + F1 \times F2$$

А разнополых:

$$p2 = 1 - p1$$

Из условия задачи следует, что

$$p2 / p1 = 11 / 5, \text{ то есть}$$

$$5 * p2 = 11 * p1 \rightarrow 5 * (1 - p1) = 11 * p1$$

Поскольку число самок во втором магазине может быть от 0 до 8, то легко проверить в цикле *for* все значения и выбрать нужное:

```
#include <iostream>
#include<windows.h>

using namespace std;

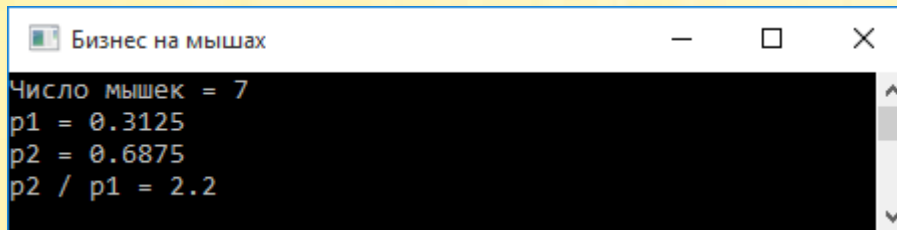
int main()
{
    SetConsoleOutputCP(1251);
    system("title Бизнес на мышах");

    for (int f = 0; f <= 8; ++f)
    {
        double M1 = 6 / 8.0;
        double F1 = 2 / 8.0;
        double M2 = (8 - f) / 8.0;
        double F2 = f / 8.0;
        double p1 = M1 * M2 + F1 * F2;

        if (5 * (1 - p1) == 11 * p1)
        {
            cout << "Число мышек = " << f << endl;
            cout << "p1 = " << p1 << endl;
            cout << "p2 = " << (1 - p1) << endl;
            cout << "p2 / p1 = " << (1 - p1) / p1 << endl;
        }
    }
}
```



```
    }  
  }  
  cout << endl;  
  
  return 0;  
}
```



```
Бизнес на мышках  
Число мышек = 7  
p1 = 0.3125  
p2 = 0.6875  
p2 / p1 = 2.2
```

Таким образом, во втором магазине было 7 самок и 1 самец.

Безусловно, эту задачу можно решить и без помощи компьютера, но она очень поучительна, если иметь в виду вещественные типы данных: стоит только забыть о десятичной точке в числовом литерале 8.0, как задача не будет решена!

Гимнастический зал

Задача из книги *Увлекательная математика*:

В гимнастическом зале стоит несколько одинаковых по длине скамей. Если спортсмены попытаются сесть по 6 человек на скамью, то одна скамья окажется незаполненной: на ней сядут лишь 3 спортсмена. Если же спортсмены попытаются сесть по 5 человек на скамью, то 4 спортсменам места не хватит.



Сколько спортсменов и сколько скамей в гимнастическом зале?

Пусть x – число спортсменов, а y – число скамей.

В первом случае x спортсменов усядутся по 6 человек на $(y - 1)$ скамью, а на одну оставшуюся – ещё трое:

$$6(y - 1) + 3 = x \quad (1)$$

Во втором случае x спортсменов усядутся по 5 человек на y скамей, и ещё четверым места не хватит:

$$5y + 4 = x \quad (2)$$

Получили простейшую **систему из двух уравнений**, которая решается перебором целых значений в бесконечном цикле *for*. Это вполне возможно, поскольку и число скамей, и число спортсменов выражаются **целыми** числами:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Гимнастический зал");

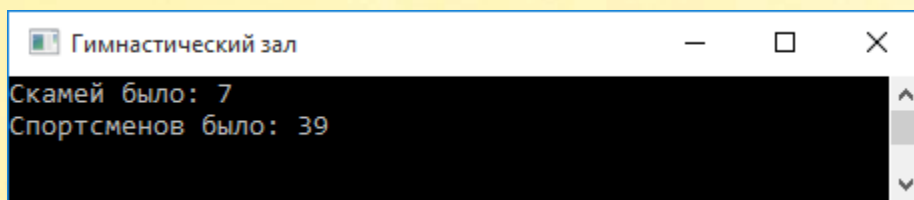
    for (int skamey = 1; ; ++skamey)
    {
        int uslovie1 = 6 * (skamey - 1) + 3;
        int uslovie2 = 5 * skamey + 4;

        if (uslovie1 == uslovie2)
        {
            cout << "Скамей было: " << skamey << endl;
            cout << "Спортсменов было: " << uslovie1 << endl;
            break;
        }
    }
}
```

```
    cout << endl;  
  
    return 0;  
}
```

Под **условиями** мы понимаем здесь правую часть уравнений (1) и (2), то есть число спортсменов.

Ответ на задачу такой:



Грузовые машины

Задача из книги *Увлекательная математика*:

Двумя грузовыми машинами требуется перевезти 143 т сыра. Грузоподъемность одной машины в 1,5 раза больше, чем другой. Для перевозки всего груза полностью гружёной машине меньшей грузоподъемности понадобится совершить 31 рейс и 27 рейсов машине большей грузоподъемности.

Сколько тонн сыра перевозит за 1 рейс каждая машина?



Обозначим буквой x грузоподъёмность первой машины, а буквой y – второй.

Тогда: $y = 1.5x$

Первая машина сделала 31 рейс и перевезла $31x$ тонн сыра.

Вторая машина сделала 27 рейсов и перевезла $27y$ тонн сыра.

Обе машины вместе перевезли 143 тонны сыра:

$$31x + 27y = 143$$

Мы получили элементарную **систему из двух линейных уравнений**, с которой элементарно справился бы и доктор Ватсон:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Грузовые машины");

    for (int mashina1 = 1; ; ++mashina1)
    {
        double mashina2 = 1.5 * mashina1;

        if (31 * mashina1 + 27 * mashina2 == 143)
        {
            cout << "Грузоподъёмность первой машины: " <<
                mashina1 << endl;
            cout << "Грузоподъёмность второй машины: " <<
                mashina2 << endl;
            break;
        }
    }
    cout << endl;
```

```
} return 0;
```

Грузоподъёмность найдена:

```
Грузовые машины
Грузоподъёмность первой машины: 2
Грузоподъёмность второй машины: 3
```

Кувшин

Задача 22 из книги *Математический фольклор*:

Четыре чашки и один кувшин для воды весят столько, сколько 17 свинцовых шариков. Кувшин весит столько, сколько одна чашка и 7 шариков.

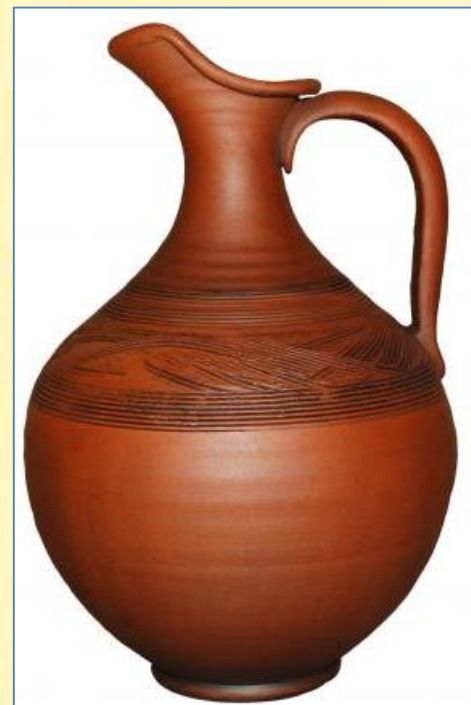
Сколько шариков уравнивают кувшин?

Вполне естественно обозначить вес кувшина в свинцовых шариках через x . Тогда на долю чашки достанется y шариков.

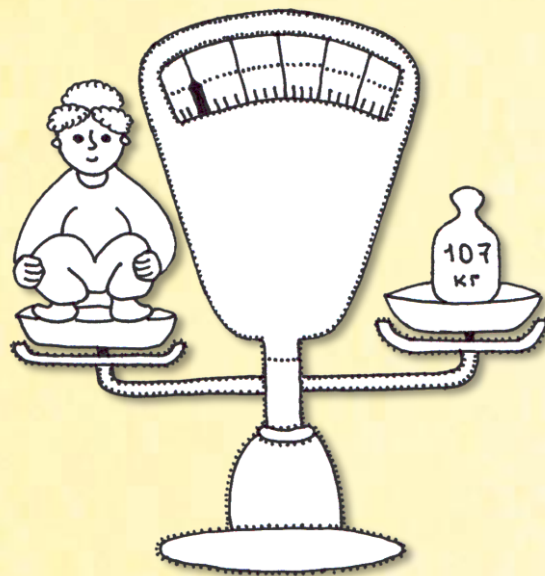
По условию задачи:

$$x = y + 7$$

$$x + 4y = 17$$



В программе вместо переменной *x* мы используем переменную **kuvshin**, а вместо *y* - **chashka**. В бесконечном цикле *for* мы изменяем вес чашки, находим вес кувшина и проверяем выполнение условия задачи:



```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Кувшин");

    for (int chashka = 1; ; ++chashka)
    {
        int kuvshin = chashka + 7;

        if (kuvshin + 4 * chashka == 17)
        {
            cout << "Вес кувшина: " << kuvshin << endl;
            cout << "Вес чашки: " << chashka << endl;
            break;
        }
    }
    cout << endl;

    return 0;
}
```

Если измерять вес не в килограммах, а в свинцовых шариках, то **ответ** на задачу такой:



Вспоминается фраза из весёлого мультика: «а в попугаях-то я гораздо длиннее!».

Ещё один кувшин

Задача 24 из книги *Математический фольклор*:

Пусть 2 чашки и 2 кувшина весят столько, сколько 14 блюдец. Один кувшин весит столько, сколько 1 чашка и 1 блюдец.

Сколько блюдец уравновесят 1 кувшин?



Чтобы нас слегка запутать и смутить, вес кувшина измеряется в этой задачке не в шариках, а в блюдцах.

Пусть кувшин весит x блюдец, а чашка - y блюдец. Тогда:

$$x = y + 1$$

$$2y + 2x = 14$$

«Перефразируя» предыдущую кувшинную задачу, получаем:

```
#include <iostream>
#include<windows.h>

using namespace std;

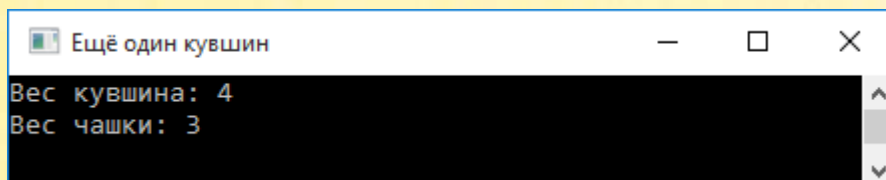
int main()
{
    SetConsoleOutputCP(1251);
    system("title Ещё один кувшин");

    for (int chashka = 1; ; ++chashka)
    {
        int kuvshin = chashka + 1;

        if (2 * kuvshin + 2 * chashka == 14)
        {
            cout << "Вес кувшина: " << kuvshin << endl;
            cout << "Вес чашки: " << chashka << endl;
            break;
        }
    }
    cout << endl;

    return 0;
}
```

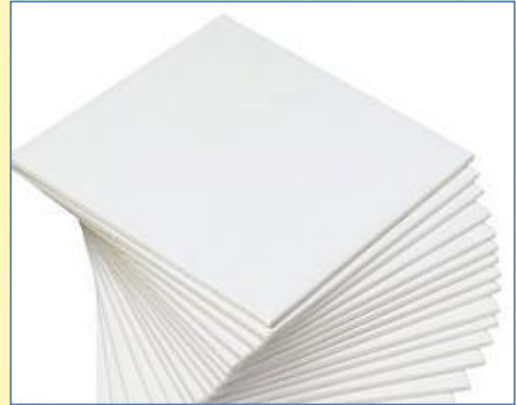
Вес посуды в блюдах такой:



```
Ещё один кувшин
Вес кувшина: 4
Вес чашки: 3
```


Складывание бумаги

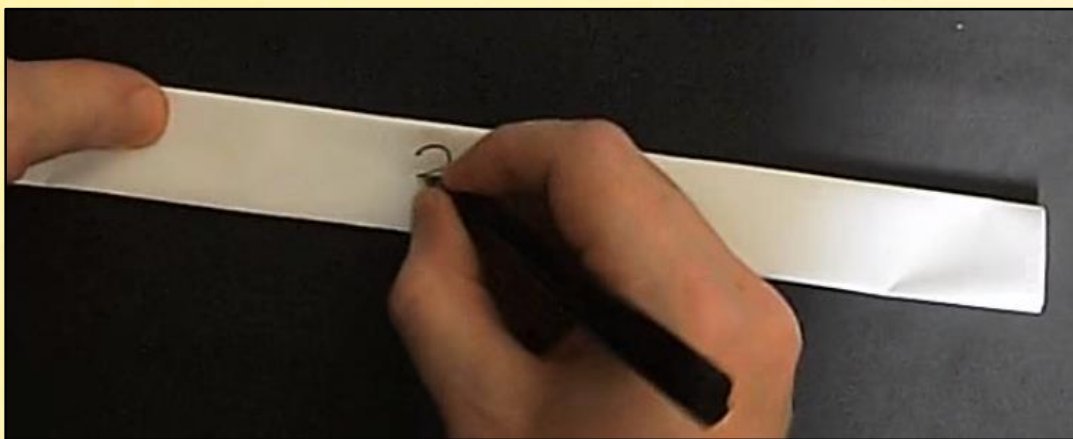
Как иногда удивительным образом стародавние задачи появляются в новом облике! Вы, конечно, помните знаменитую шахматную задачу. Казалось бы, вполне современная головоломка со складыванием листа бумаги не имеет с ней родственных связей, но это не так. Обе поражающие воображение задачи основаны на свойствах показательной функции.



В 28-ом математическом выпуске *Академии занимательных наук* профессор Круглов наглядно доказывает хомячку Циркулю, что лист бумаги формата А4 нельзя сложить вдвое более **6 раз**:



Но в *Интернете* можно найти ролик, в котором показано, как такой же лист бумаги можно сложить **7 раз**:



Фокус в том, что сначала листок нужно складывать по длинной стороне, а не попеременно по длинной – по короткой.

Нашлись умельцы, которые взяли огромный лист и свернули его **9 раз**:



Давайте разбираться, почему бумага так упорно не желает складываться вдвое!

Так как мы не проводим натурные испытания, то довольствуемся простой **математической моделью** процесса складывания бумаги вдоль и поперёк. Она несущественно упрощает этот процесс, зато описывается элементарной формулой:

$$d = a \cdot 2^n$$

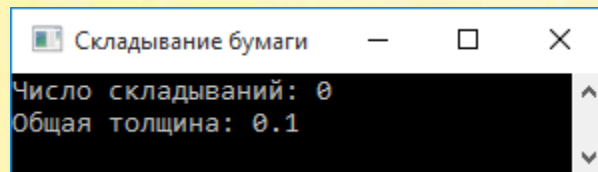
Здесь:

d – общая толщина сложенного листа бумаги

a – толщина самого листа

n – число складываний

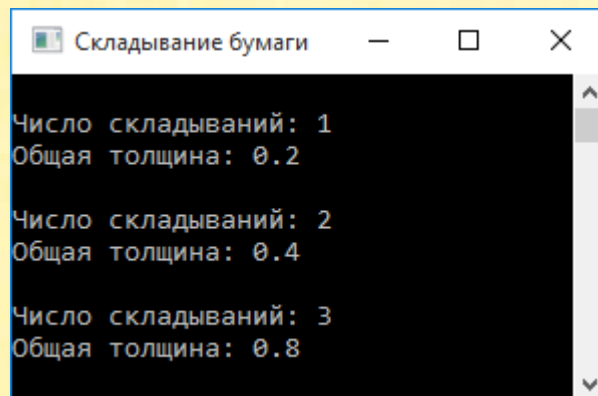
Возьмём достаточно тонкую бумагу толщиной **a = 0.1** мм. В начале эксперимента весь «свёрток» имеет такую же толщину:



```
Складывание бумаги
Число складываний: 0
Общая толщина: 0.1
```

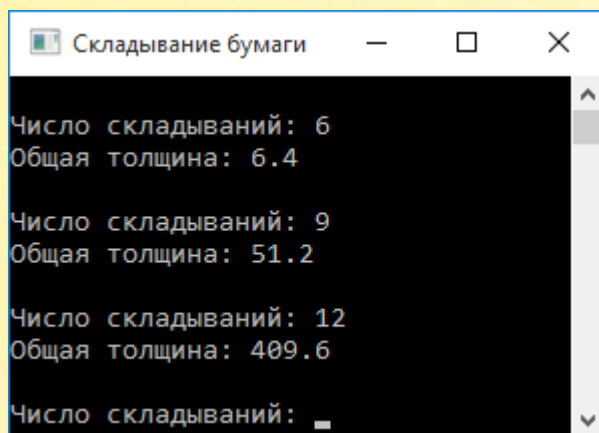
Пока всё нормально.

После трёх складываний толщина сложенного листа бумаги всё ещё меньше 1 миллиметра:



```
Складывание бумаги
Число складываний: 1
Общая толщина: 0.2
Число складываний: 2
Общая толщина: 0.4
Число складываний: 3
Общая толщина: 0.8
```

Но при последующих складываниях толщина листа стремительно растёт:



```
Складывание бумаги
Число складываний: 6
Общая толщина: 6.4

Число складываний: 9
Общая толщина: 51.2

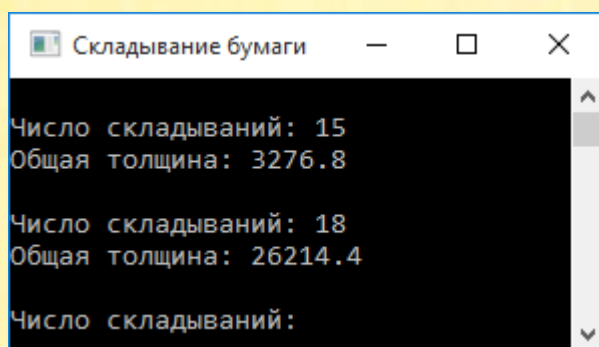
Число складываний: 12
Общая толщина: 409.6

Число складываний: █
```

После 9 складываний толщина достигнет 5 сантиметров, а после 12 – почти 41 сантиметра. Теперь легко представить размеры исходного листа бумаги, чтобы его можно было сложить пополам при такой толщине.

Если конечная площадь верхней поверхности листа 40 квадратных сантиметров, то исходный лист при 12 складываниях должен быть в 4096 раз больше, то есть 163840 квадратных сантиметров, или 16,4 квадратных метра. На самом деле площадь исходного листа должна быть значительно больше, поскольку мы складываем не отдельные листы стопкой, а единственный лист, часть которого образует боковые поверхности. Также мы предполагаем, что лист размером 40 x 80 сантиметров толщиной 40 сантиметров ещё можно сложить вдвое.

Но если 12 раз сложить вдвое огромный лист тонкой (!) бумаги ещё вполне возможно, то дальше толщина листа будет измеряться метрами:



```
Складывание бумаги
Число складываний: 15
Общая толщина: 3276.8

Число складываний: 18
Общая толщина: 26214.4

Число складываний: █
```

Здесь мы наблюдаем ту же самую картину, что и при выкладывании зёрен на шахматную доску. Толщина листа бумаги растёт так быстро, что даже 15 складываний проделать не удастся.

В книге Вальтера Литцмана *Великаны и карлики в мире чисел*, пятое издание которой вышло в Лейпциге в 1953 году, эксперимент со складыванием бумаги приводится как пример быстрого роста показательной функции. При складывании листа бумаги толщиной $1/10$ мм 40 раз высота «стопки» превысит 100 000 километров!

А вот совсем короткая **программа**, которая помогла нам провести «мысленный» эксперимент:

```
#include <iostream>
#include<windows.h>
#include<math.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Складывание бумаги")

    double tolshinaLista = 0.1;
    int nSklad;
    while(true)
    {
        cout << "Число складываний: ";
        cin >> nSklad;
        double tolshina = tolshinaLista * pow(2, nSklad);
        cout << "Общая толщина: " << tolshina << endl;
        cout << endl;
    }

    return 0;
}
```



Bogenschießen

В задаче из немецкой книги *Gehirnjogging* речь идёт о стрельбе из лука по мишени, но мы легко найдём в ней диофантово уравнение.



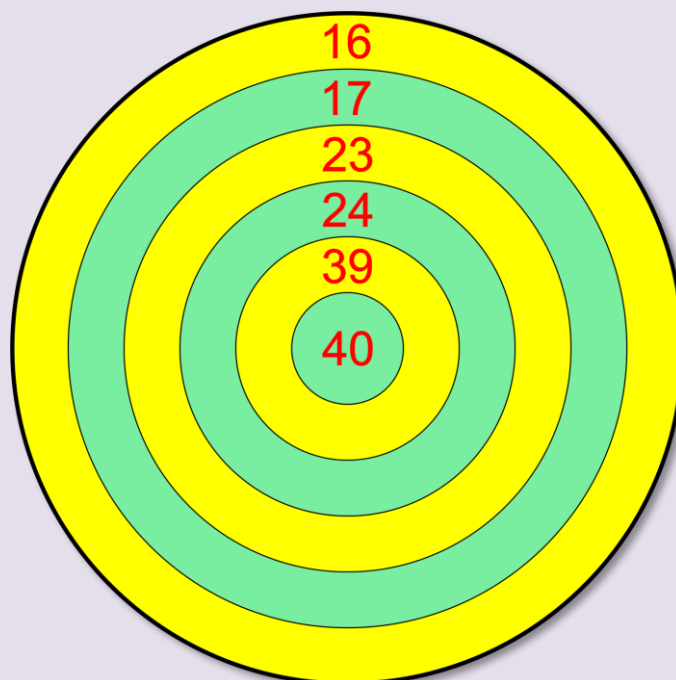
Задача 26. Bogenschießen. Вот её условие на немецком языке:

Lilly hat im Garten eine Zielscheibe aufgebaut.

*Welche Felder muss sie treffen, um genau auf 100 Punkte zu kommen?
(Sie darf beliebig viele Pfeile abschießen.)*

Стрельба из лука

У Лили в саду висит мишень:



*Как Лилиа может выбить ровно 100 очков?
(Количество стрел может быть любым.)*

C++ не переносит немецкую букву β (эцет), поэтому название проекта пришлось изменить.

На оригинальной мишени число 24 встречается дважды, что уже совсем нехорошо. Пришлось заменить его числом 23.

Задача решается простым перебором во вложенных циклах *for*:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Gehirnjogging. Задача 26 ");

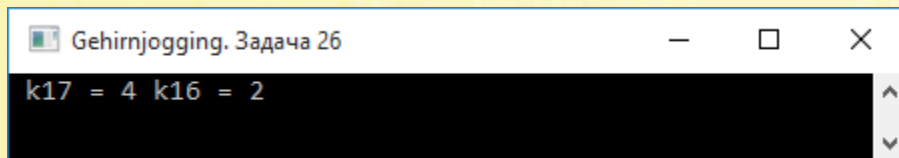
    const int SCORE = 100;
    const int C40 = SCORE / 40;
    const int C39 = SCORE / 39;
    const int C24 = SCORE / 24;
    const int C23 = SCORE / 23;
    const int C17 = SCORE / 17;
    const int C16 = SCORE / 16;

    for (int k40 = 0; k40 <= C40; ++k40)
        for (int k39 = 0; k39 <= C39; ++k39)
            for (int k24 = 0; k24 <= C24; ++k24)
                for (int k23 = 0; k23 <= C23; ++k23)
                    for (int k17 = 0; k17 <= C17; ++k17)
                        for (int k16 = 0; k16 <= C16; ++k16)
                        {
                            if (k40 * 40 + k39 * 39 +
                                k24 * 24 + k23 * 23 +
                                k17 * 17 + k16 * 16 == SCORE)
                            {
                                if (k40 > 0)
```

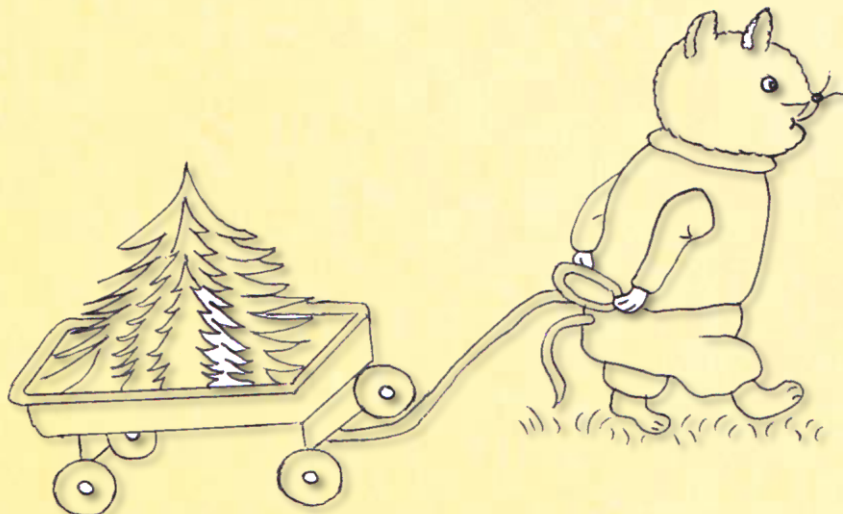
```

        cout << " k40 = " << k40;
    if (k39 > 0)
        cout << " k39 = " << k39;
    if (k24 > 0)
        cout << " k24 = " << k24;
    if (k23 > 0)
        cout << " k23 = " << k23;
    if (k17 > 0)
        cout << " k17 = " << k17;
    if (k16 > 0)
        cout << " k16 = " << k16;
    cout << endl;
}
}
cout << endl;
return 0;
}

```



Если наверняка знать, что ответ единственный, то задачу можно решить и в уме.



Город

В журнале *Квантик*, №1 за 2016 год опубликованы задачи Первого тура математического конкурса. Мы решим **первую задачу**:



«Докажите» здесь значит «решите за-

1. Город разделён рекой на две половины, в каждой половине живёт по миллиону человек. В первый год 2015 человек переселились из левой половины в правую; во второй год 2016 человек переселились из правой половины в левую; в третий год опять 2015 человек переселились слева направо; в четвёртый год – 2016 человек переселились справа налево, и так далее.

Докажите, что в какой-то год в каждой из половин снова окажется по миллиону жителей. Через сколько лет это случится?

дачу». Задача очень простая, но не сразу это сообразишь, поэтому мы напишем короткую программу, которая выдаст нам правильный ответ, который покажет, как можно решить эту задачу исключительно в уме.

Главная функция начинается с объявления **переменных**, известных вам по условию задачи:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Квантик №1 за 2016");

    // население:
    int gorod1 = 1000000;
    int gorod2 = 1000000;
    // переселение:
```

```
int iz1 = 2015;
int iz2 = 2016;
```

А дальше мы моделируем процесс переселения населения с насиженных мест в другую часть города.

Поскольку число переселений нам неизвестно (иначе мы бы знали ответ на задачу), то цикл **while** будет бесконечным, а прервём мы его тогда, когда выполнится требование задачи: в каждой половине города вновь окажется по 1 миллиону жителей.

В первый год, а также во все последующие нечётные года **iz1** человек переселяется из одной половины города в другую. Во второй год и во все чётные года, наоборот, из второй половины города переселяется **iz2** человек в первую.

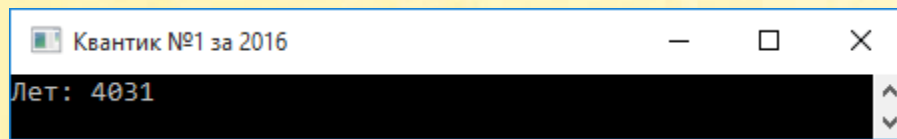
Через **n** лет население обеих половин города сравняется, и мы напечатаем ответ:

```
// счётчик лет:
int n = 0;
while (true)
{
    n += 1;
    if (n % 2) // из первого во второй
    {
        gorod1 -= iz1;
        gorod2 += iz1;
    }
    else // # из второго в первый
    {
        gorod1 += iz2;
        gorod2 -= iz2;
    }

    // проверяем численность:
    if (gorod1 == 1000000)
    {
        cout << "Лет: " << n << endl;
        break;
    }
}
```

```
    }  
  }  
  cout << endl;  
  
  return 0;  
}
```

Запускаем программу и тут же узнаём, что долгожданное событие наступит через 4031 год:



Если бы найти такой город, в котором люди живут по 4000 лет, то можно было бы и переселяться ежегодно.

Понятно, что эта задача совершенно искусственная, а число переселенцев просто обозначает предыдущий и текущий года.

Ответ 4031 нетрудно получить так: $2015 \times 2 + 1$. Откуда и следует элементарное решение.

Каждые 2 года население правой части города уменьшается на 1 человека. Но при этом из левой части города каждый нечётный год к ним добавляются 2015 человек. Когда в правой части останется $1000000 - 2015$ человек, на следующий год в него придут 2015 человек из левой части города, и население возрастёт до 1000000 человек. Это будет как раз в тот год, о котором и спрашивается в условии задачи.

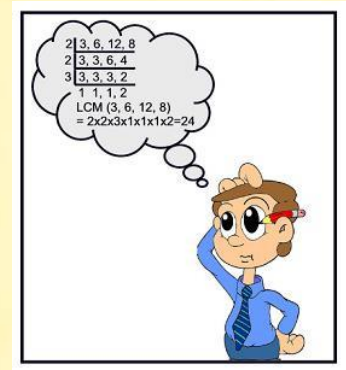
Как мы уже выяснили, население правой части города уменьшается на 1 человека за 2 года, поэтому на 2015 человек население уменьшится за $2015 \times 2 = 4030$ лет. А на следующий, 4031-й год 2015 человек из левой части города увеличат его до 1 миллиона.

Хорошее дело – программирование, но иногда и мозгами нелишне пораскинуть!

Наименьшее число

В журнале *Наука и жизнь*, №2 за 1968 год, на странице 57 напечатана такая задача:

**НАЙТИ
НАИМЕНЬШЕЕ ЧИСЛО**
Найдите наименьшее число, которое при делении на 2, 3, 4, 5 и 6 дает остатки соответственно 1, 2, 3, 4 и 5.



Задача исключительно на сообразительность. Вот журнальное решение задачи:

**НАЙТИ НАИМЕНЬШЕЕ
ЧИСЛО**
Прибавим к искомому числу 1. Полученная сумма должна делиться одновременно на 2, 3, 4, 5 и 6. Наименьшее такое число 60. Следовательно, искомое число:

60 — 1 = 59.

Не каждый этим похвальным даром обладает, поэтому мы пишем простенькую программу:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
```

```

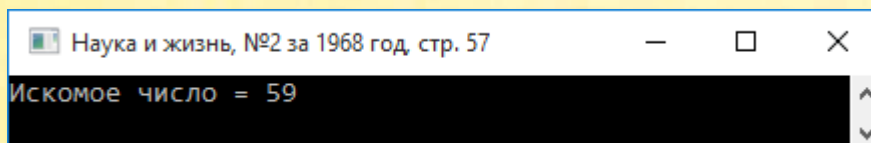
system("title Наука и жизнь, №2 за 1968 год, стр. 57");

int n = 0;
while (true)
{
    n += 1;
    bool flg = true;
    for (int i = 2; i <= 6; ++i)
    {
        if (n % i != i - 1)
        {
            flg = false;
            break;
        }
    }
    if (flg)
    {
        cout << "Искомое число = " << n;
        break;
    }
}
cout << endl;

return 0;
}

```

Когда искомое число заведомо небольшое, то его легко найти простым перебором в бесконечном цикле *while*. Он заканчивается, когда найдено первое число, удовлетворяющее всем условиям задачи:



The screenshot shows a standard Windows command prompt window. The title bar reads "Наука и жизнь, №2 за 1968 год, стр. 57". The main area of the window is black with white text that says "Искомое число = 59". There are standard window control buttons (minimize, maximize, close) in the top right corner.

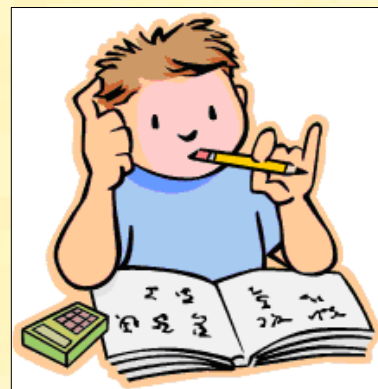
Вопреки ожиданиям наш ответ полностью совпал с «сообразительным» ...

Трёхзначное число

В книге *600 задач на сообразительность*, на странице 112 напечатана задача 41:

Трёхзначное число

Если от трёхзначного числа отнять 7, то оно разделится на 7; если отнять от него 8, то оно разделится на 8; если отнять от него 9, то оно разделится на 9. Какое это число?



Опять задача на сообразительность, которой мы как бы не обладаем, а потому сразу садимся за компьютер и решаем задачу:

```
#include <iostream>
#include<windows.h>

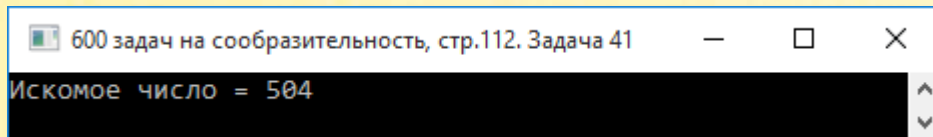
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title 600 задач на сообразительность, стр.112. Задача 41");

    int n = 0;
    while (true)
    {
        n += 1;
        if ((n - 7) % 7)
            continue;
        if ((n - 8) % 8)
            continue;
        if ((n - 9) % 9)
            continue;
        break;
    }
    cout << "Искомое число = " << n;
```

```
    cout << endl;  
    return 0;  
}
```

Ответ такой:



```
600 задач на сообразительность, стр.112. Задача 41  
Искомое число = 504
```

Он, безусловно, правильный, но, чтобы его получить, вполне можно обойтись и смекалкой.

Действительно, если число $n - 7$ нацело делится на 7, то и число n кратно 7. Аналогично для чисел 8 и 9. Стало быть искомое число одновременно делится на 7, 8 и 9.

Минимальное число, обладающее такими свойствами равно:

$$7 * 8 * 9 = 504$$

Следующее такое число $504 * 2 = 1008$, но оно не трёхзначное.

Треугольные числа

Треугольные числа – это частный вид *фигурных чисел*.

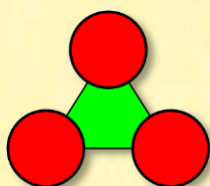
Если из определённого числа кружков можно построить правильный треугольник, то такое число и называют треугольным.

Самое **первое** треугольное число – это единица. Правда, один кружок не очень-то похож на треугольник, но у математиков свой взгляд на вещи.

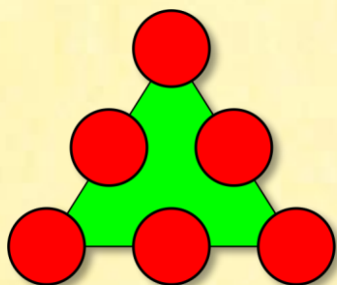


Следует добавить, что **нулевое** треугольное число представляет собой треугольник, которого вообще не видно. Он существует только в мозгу математиков.

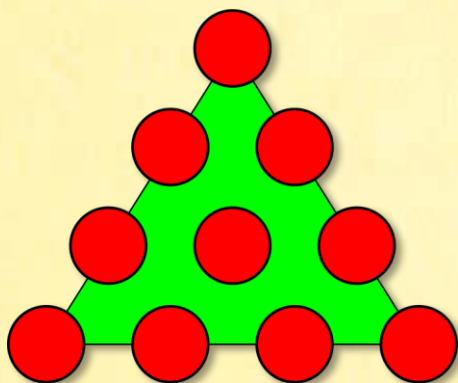
Второе треугольное число – тройка. Из трёх кружочков уже можно построить вполне реалистичный треугольник:



Третье треугольное число – шестёрка:



Четвёртое треугольное число – десятка:



Дальше можно не продолжать – закономерность понятна:

Первое число равно 1.
Второе – 1 + 2.
Третье - 1 + 2 + 3.
Четвёртое - 1 + 2 + 3 + 4.
Пятое - 1 + 2 + 3 + 4 + 5.

n -ное треугольное число равно сумме первых n натуральных чисел.

Ряд треугольных чисел представляет собой простейшую арифметическую прогрессию, поэтому любое треугольное число можно легко вычислить по **формуле**:

$$T_n = \frac{1}{2} n(n + 1) \quad (1)$$

Однако из наших нарисованных треугольников видно, что второе треугольное число получается, если к первому прибавить 2. Третье – если ко второму прибавить 3. Для любого треугольного числа верна такая **рекуррентная формула**:

$$T_n = T_{n-1} + n$$

Из неё хорошо видна прямая аналогия с рекурсивным вычислением факториалов. Нам нужно только заменить умножение сложением:

```
#include <iostream>
#include<windows.h>

using namespace std;

// Рекурсивный способ вычисления треугольных чисел

// РЕКУРСИВНАЯ ФУНКЦИЯ
int tri(int n)
{
    if (n == 0)
        return 0;
```

```

    return n + tri(n-1);
}

int main()
{
    SetConsoleOutputCP(1251);
    system("title Треугольные числа");

    cout << endl;
    for (int n = 0; n < 11; ++n)
        cout << "Треугольное число " << n << " = " << tri(n) << endl;
    cout << endl;

    return 0;
}

```

```

Треугольные числа
Треугольное число 0 = 0
Треугольное число 1 = 1
Треугольное число 2 = 3
Треугольное число 3 = 6
Треугольное число 4 = 10
Треугольное число 5 = 15
Треугольное число 6 = 21
Треугольное число 7 = 28
Треугольное число 8 = 36
Треугольное число 9 = 45
Треугольное число 10 = 55

```

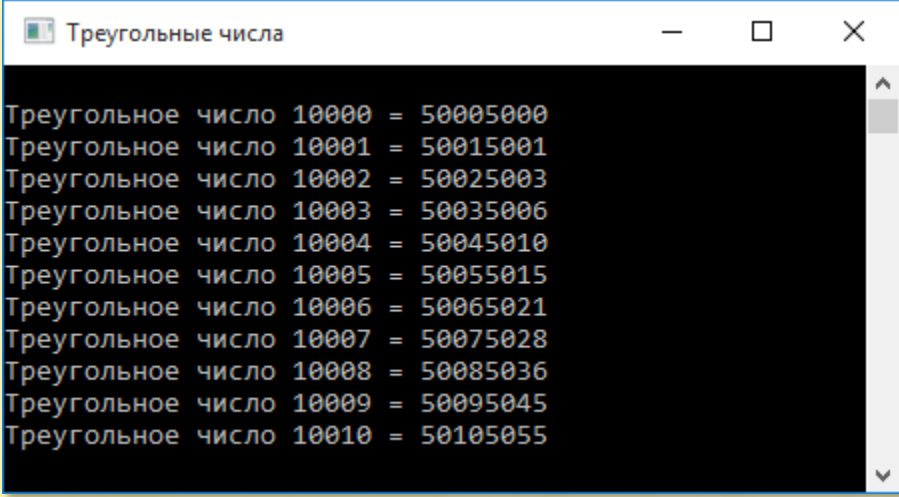
Если вам вдруг понадобятся очень большие треугольные числа, то их можно получить от **нерекурсивной функции**:

```

// НЕРЕКУРСИВНАЯ ФУНКЦИЯ
unsigned long long tri2(int n)
{
    if (n <= 1)
        return n;
}

```

```
unsigned long long t = n * (n + 1) / 2;  
return t;  
}
```

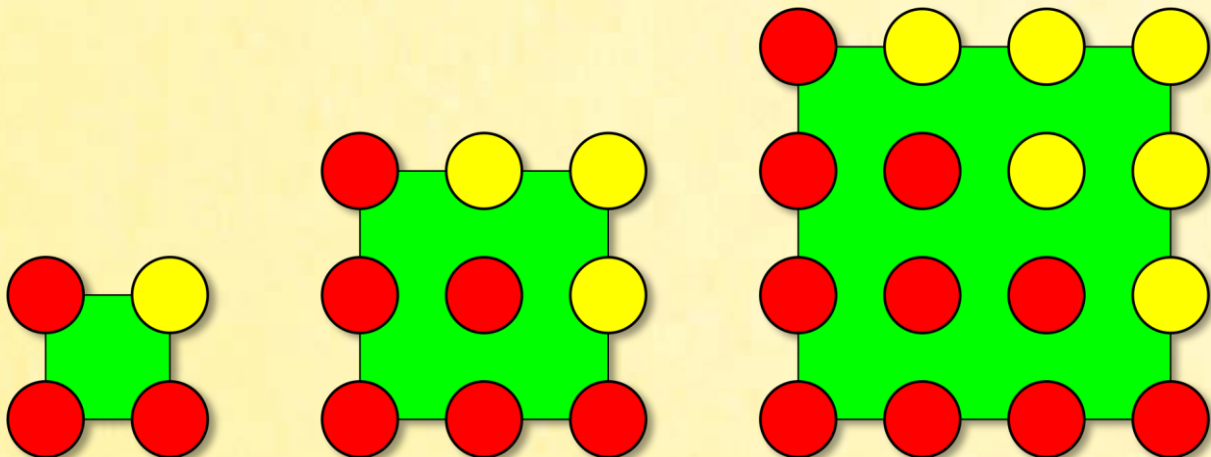


```
Треугольное число 10000 = 50005000  
Треугольное число 10001 = 50015001  
Треугольное число 10002 = 50025003  
Треугольное число 10003 = 50035006  
Треугольное число 10004 = 50045010  
Треугольное число 10005 = 50055015  
Треугольное число 10006 = 50065021  
Треугольное число 10007 = 50075028  
Треугольное число 10008 = 50085036  
Треугольное число 10009 = 50095045  
Треугольное число 10010 = 50105055
```

Интересно, что сумма двух последовательных треугольных чисел - это **квадратное число**:

$$T_n + T_{n+1} = (n+1)^2$$

На рисунке это свойство хорошо видно:

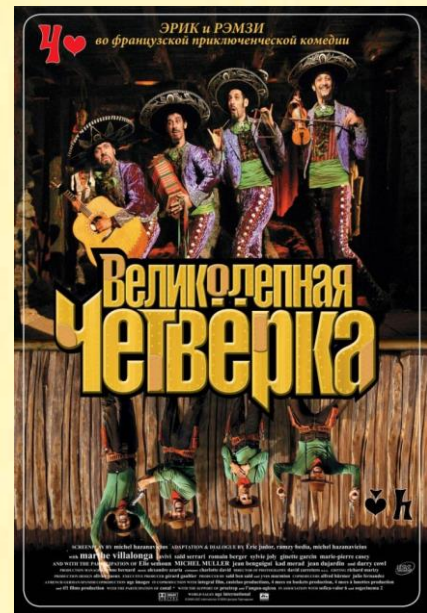
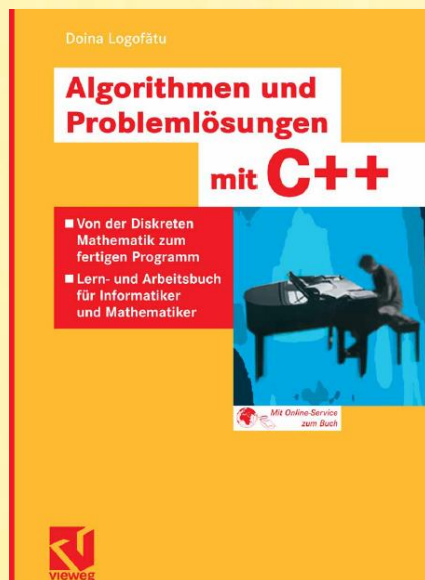


А точное доказательство этого утверждения легко получить из формулы (1), если найти сумму чисел T_n и T_{n+1} .

Менее очевидно такое свойство треугольных чисел: *любое целое неотрицательное число можно представить в виде суммы не более трёх треугольных чисел*. Это предположение высказал Пьер Ферма в 1638 году, а доказал его в 1796 году Карл Гаусс.

Великолепная четвёрка

В книге *Algorithmen und Problemlösungen mit C++*, на страницах 300-302 решается такая задача.



Их четвёрки с помощью трёх простых операций получить любое натуральное число.

Операции такие:

- добавить 4 в конец
- добавить 0 в конец
- разделить число на 2, если оно чётное

Эту задачу проще решить с **конца**, то есть заданное число превратить в четвёрку. Тогда и операции станут противоположными:

- удалить 4, если она стоит в конце числа
- удалить 0, если он стоит в конце числа
- умножить число на 2

В **главной функции** мы вводим любое целое число. Если это нуль или отрицательное число, то программа закрывается. В противном случае число *num* передаётся функции **solve**. Кроме того, она получает сообщения, которые поясняют действие функции.

Если число *num* - **четвёрка** (основной случай), то решение заканчивается. В противном случае мы рекурсивно выполняем операции над текущим числом *num*.

Если число *num* **заканчивается на нуль**, то мы удаляем его. Для этого достаточно разделить число на 10.

Аналогично мы поступаем, если число **заканчивается на четвёрку**.

Если число **заканчивается на другие цифры**, то мы умножаем его на 2.

Эти действия мы продолжаем до тех пор, пока число не обратится в четвёрку. Тогда мы печатаем все действия с комментариями в обратном порядке – по мере возврата из вызванных функций.

Обратите внимание, что в функции **solve** операции противоположные, а комментарии «прямые», так как функция печатает все действия в прямом порядке.

```
#include <iostream>
#include<windows.h>

using namespace std;

// РЕШАЕМ ЗАДАЧУ
void solve(int num, string message)
{
    if (num == 4)
    {
        cout << num;
```

```

        return;
    }

    // последняя цифра числа:
    int lastDig = num % 10;
    // добавляем 0 в конец:
    if (lastDig == 0)
        solve(num / 10, "(добавляем 0 в конец)");
    // добавляем 4 в конец:
    else if (lastDig == 4)
        solve(num / 10, "(добавляем 4 в конец)");
    // делим на 2:
    else
        solve(num * 2, "(делим на 2)");

    cout << " -> " << num << message;
}

int main()
{
    SetConsoleOutputCP(1251);
    system("title Algorithmen und Problemlösungen mit C++, стр. 300-302");

    // бесконечный цикл ввода данных -
    // пока пользователь не закроет программу
    // или не введёт нуль или отрицательное число:
    while (true)
    {
        cout << "Введите число > ";
        int num;
        cin >> num;
        // если пользователь ввёл 0 или отрицательное число,
        // то программу закрываем:
        if (num <= 0)
            return 0;
        // решаем задачу:
        solve(num, "");
        cout << endl;
        cout << endl;
    }
}

```

На рисунке вы видите, как работает наша программа:

```
Algorithmen und Problemlösungen mit C++, стр. 300-302
Введите число > 1
4 -> 2(делим на 2) -> 1

Введите число > 3
4 -> 2(добавляем 4 в конец) -> 24(делим на 2) -> 12(делим на 2) -> 6(делим на 2) -> 3

Введите число > 7
4 -> 2(делим на 2) -> 1(добавляем 4 в конец) -> 14(делим на 2) -> 7

Введите число > 1000
4 -> 2(делим на 2) -> 1(добавляем 0 в конец) -> 10(добавляем 0 в конец) -> 100(добавляем 0 в конец) -> 1000

Введите число > 2016
4 -> 2(добавляем 4 в конец) -> 24(делим на 2) -> 12(делим на 2) -> 6(добавляем 4 в конец) -> 64(добавляем 4 в конец) -> 644(делим на 2) -> 322(добавляем 4 в конец) -> 3224(делим на 2) -> 1612(делим на 2) -> 806(добавляем 4 в конец) -> 8064(делим на 2) -> 4032(делим на 2) -> 2016

Введите число >
```

Рекурсивный тортик

В книге *Математические изюминки* [ХР92], на страницах 9-11 Росс Хонсбергер показывает, как решить такую задачу:

Расположим n точек на окружности и соединим их попарно хордами. Предположим, что никакие три хорды не имеют общей точки внутри круга.

На сколько областей разобьётся круг этими хордами?



Вывод формулы вы можете проследить по книге, нас же интересует только формула:

$$NR(n) = C_n^2 + C_n^4 + 1$$

Здесь:

- NR – число областей, на которые хорды разбивают круг
- n – число точек на окружности

$$C_n^2$$

$$C_n^4$$

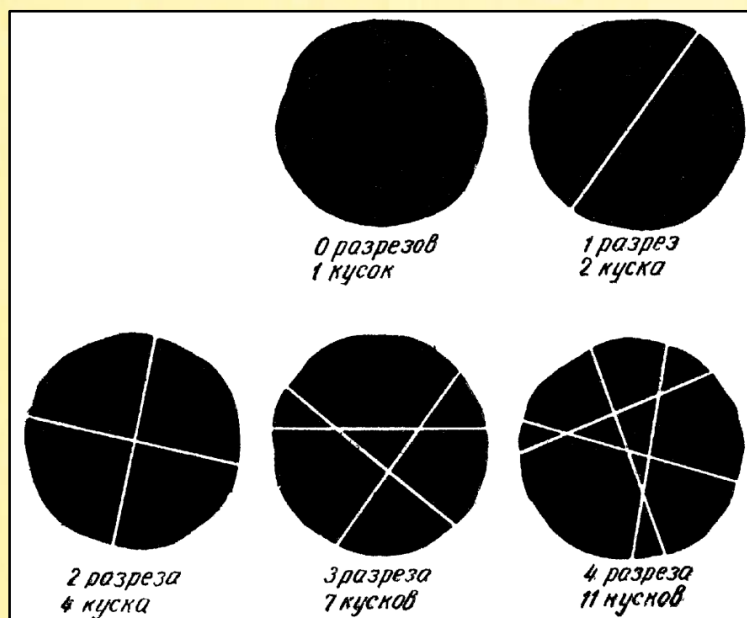
- C_n^2 и C_n^4 – число сочетаний по 2 и по 4 из n

Подобную задачу решает и Мартин Гарднер в книге **Математические досуги** [ГМ72], на страницах 82-87.

Она облечена в более занимательную форму:

На какое максимальное число кусков можно разделить круглый пирог, если сделать n разрезов, каждый из которых пересекает все остальные?

На рисунке вы можете внимательно проследить судьбу торта, подвергнутого означенным разрезам.



По этим данным с помощью *метода конечных разностей* легко выводится **формула** для подсчёта кусков:

$$NR = \frac{1}{2} n^2 + \frac{1}{2} n + 1 = \frac{1}{2} n(n + 1) + 1 \quad (1)$$

Существует и **рекуррентная формула**:

$$NR(0) = 1$$
$$NR(n) = n + NR(n-1) \text{ при } n > 0$$

Давайте в помощь домохозяйкам и кондитерам напишем программу, умеющую подсчитывать куски торта, которые получаются при заданном числе разрезов.

Поскольку в реальной жизни вряд ли придётся разрезать даже самый большой торт на тысячи частей (то есть в пыль!), то для практических нужд вполне достаточно **рекуррентной формулы**:

```
#include <iostream>
#include<windows.h>

using namespace std;

// РЕЖЕМ ТОРТИК РЕКУРСИВНО
int TortikRec(int num)
{
    if(num < 1)
        return 1;
    else
        return num + TortikRec(num - 1);
}

int main()
{
    SetConsoleOutputCP(1251);
    system("title Математические изюминки, стр. 9-11");
```

```

// бесконечный цикл ввода данных -
// пока пользователь не закроет программу
// или не введёт нуль или отрицательное число:
while (true)
{
    cout << "Введите число разрезов > ";
    int num;
    cin >> num;
    // если пользователь ввёл отрицательное число,
    // то программу закрываем:
    if (num < 0)
        return 0;

    // находим число разрезов:
    num = TortikRec(num);
    // печатаем число кусков:
    cout << "Максимальное число кусков = " << num;

    cout << endl;
    cout << endl;
}
}

```

```

Математические изюминки, стр. 9-11
Введите число разрезов > 0
Максимальное число кусков = 1

Введите число разрезов > 1
Максимальное число кусков = 2

Введите число разрезов > 10
Максимальное число кусков = 56

Введите число разрезов > 100
Максимальное число кусков = 5051

Введите число разрезов >

```

Но если вас интересуют и теоретические изыскания в кондитерской науке, то нужно прибегнуть к формуле (1):

```

int Tortik(int num)
{
    int res = num*(num + 1) / 2 + 1;
    return res;
}

```

```

Математические изюминки, стр. 9-11
Введите число разрезов > 500
Максимальное число кусков = 125251

Введите число разрезов > 1000
Максимальное число кусков = 500501

Введите число разрезов > 2000
Максимальное число кусков = 2001001

Введите число разрезов > 2016
Максимальное число кусков = 2033137

Введите число разрезов > _

```

Тортик

Впрочем, хозяйке гораздо интереснее и полезнее знать, **сколько нужно сделать разрезов**, чтобы получить вполне определённое число кусков торта.

В функции `main` она задаёт нужное число кусков и получает от функции `Tortik` необходимое число разрезов:

```

#include <iostream>
#include<windows.h>

using namespace std;

int main()
{

```

```

SetConsoleOutputCP(1251);
system("title Тортик");

// бесконечный цикл ввода данных -
// пока пользователь не закроет программу
// или не введёт нуль или отрицательное число:
while (true)
{
    cout << "Введите число кусков > ";
    int num;
    cin >> num;
    // если пользователь ввёл отрицательное число,
    // то программу закрываем:
    if (num < 0)
        return 0;

    // находим число разрезов:
    int tile = Tortik(num);

    // печатаем число разрезов:
    cout << "Минимальное число разрезов = " << tile << endl;
    int unwanted = tile * (tile + 1) / 2 + 1 - num;
    cout << "Число лишних кусков = " << unwanted;

    cout << endl;
    cout << endl;
}
}

```

В функции **Tortik** нам удобнее пользоваться *нерекуррентной* формулой.

Она получает необходимое число кусков **n** и начинает последовательно делать *num* разрезов – до тех пор, пока число кусков не сравняется или не превысит заданного числа кусков:

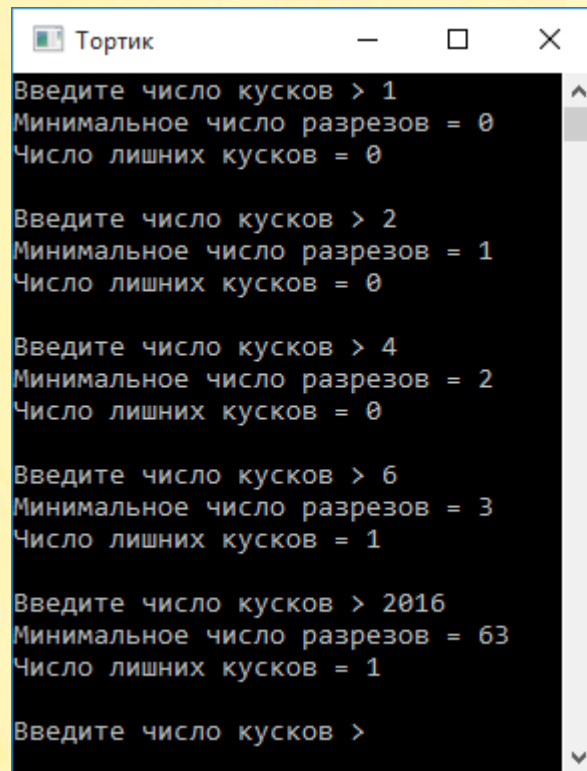
```

// РЕЖЕМ ТОРТИК
int Tortik(int num)
{
    int res = 0;

```

```
}  
while(num > (res*(res + 1) / 2 + 1))  
    res += 1;  
return res;  
}
```

На рисунке вы видите работу этой во всех отношениях полезной для дома и для семьи программы:



```
Тортик  
Введите число кусков > 1  
Минимальное число разрезов = 0  
Число лишних кусков = 0  
  
Введите число кусков > 2  
Минимальное число разрезов = 1  
Число лишних кусков = 0  
  
Введите число кусков > 4  
Минимальное число разрезов = 2  
Число лишних кусков = 0  
  
Введите число кусков > 6  
Минимальное число разрезов = 3  
Число лишних кусков = 1  
  
Введите число кусков > 2016  
Минимальное число разрезов = 63  
Число лишних кусков = 1  
  
Введите число кусков >
```

В некоторых случаях остаются «лишние» куски. Вот почему так важно знать математику и **лично** нарезать тортики кусками!

Столетие

*В кашне, ладонью заслонясь,
Сквозь фортку крикну детворе:
Какое, милые, у нас
Тысячелетье на дворе?*

Борис Леонидович Пастернак, *Про эти стихи*

Первая задача школьного этапа Всероссийской олимпиады по информатике 2016-2017 для 7-8 классов:

По четырёхзначному номеру года, запрошенного с клавиатуры, определите номер столетия (например, для 1342 года ответ XIV век, для 1918 года – XX век). Учтите, что началом века считается первый, а не нулевой год. (То есть, 2000-й год – это последний год XX века).

Так как век – это 100 лет, то номер года нужно разделить на 100. Проверим, что получается для 20-го века.

$$1901 / 100 = 19.01$$

$$2000 / 100 = 20.0$$

Вывод 1: в частном нужно отбрасывать дробную часть.

Недаром в программировании счёт начинают с нуля. Мы получили для начала века другой результат, чем для всех остальных именно потому, что люди считают с единицы, а не с нуля. Вычтем из числа года 1, тогда в 20 веке года попадут в диапазон 1900..1999. При делении нацело получим:

$$1900 / 100 = 19$$

$$1999 / 100 = 19$$

Все промежуточные года также дадут частое, равное 19. Уже хорошо, но век получается на 1 меньше, чем следует. Добавим к частному 1 – и задача решена.

Программа в бесконечном цикле *while* спрашивает у пользователя год, а в переменную *vek* записывает номер столетия:

```
#include <iostream>
#include<windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Столетие");

    // бесконечный цикл ввода данных -
    // пока пользователь не закроет программу
    // или не введёт нуль или отрицательное число:
    while (true)
    {
        cout << "Введите год > ";
        int god;
        cin >> god;
        // если пользователь ввёл отрицательное число,
        // то программу закрываем:
        if (god < 0)
            return 0;

        // находим столетие:
        int vek = (god - 1) / 100 + 1;

        // печатаем столетие:
        cout << "Столетие: " << vek << endl;
        cout << endl;
    }
}
```

Как и полагается, проверяем программу на экстремальных значениях:

```
Столетие
Введите год > 1901
Столетие: 20

Введите год > 2000
Столетие: 20

Введите год > 111
Столетие: 2

Введите год > 1
Столетие: 1

Введите год >
```

Очевидно, что все промежуточные года будут вычислены верно. То же самое относится и к другим векам. Почему номер года должен быть четырёхзначным, непонятно. Для всех натуральных чисел программа вычисляет столетие верно.

Суперпростые числа

Под суперпростыми понимают **разные** числа, но мы исследуем только такие числа, которые и целиком – простые, и все их дробные части – также простые. Для примера возьмём простое число **1373**. Разобьём его всеми возможными способами на две части:

1-373
13-73
137-3

Оказывается, что все «дробные числ»а – *1, 3, 13, 73, 137, 373* - также простые. А раз это так, то исходное число *1373* мы по полному праву можем именовать **суперпростым!**

Наша **задача** - отыскать все суперпростые числа заданной длины *len*. Так как однозначные числа нельзя поделить на части, то они формально суперпростые, но

очень неинтересные, поэтому мы разрешим пользователю задавать длину чисел в разумном диапазоне 2..7:

```
#include <iostream>
#include<windows.h>
#include<math.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Суперпростые числа");

    // бесконечный цикл ввода данных -
    // пока пользователь не закроет программу
    // или не введёт нуль или отрицательное число:
    int len;
    do
    {
        cout << "Введите число 2..7 > ";
        cin >> len;
        // если пользователь ввёл отрицательное число
        //или 0, то программу закрываем:
        if (len <= 0)
            return 0;

    }
    while((len >= 2) && (len <= 7));
}
```

По заданной длине чисел мы находим максимальное число *max*, которое на 1 больше максимального из заданных:

```
cout << endl;
//ищем суперпростые числа
//заданной длины:
int max = 1;
```

```
for (int i = 1; i <= len; ++i)
    max *= 10;
```

Например, для `len = 2` мы получим 1 с двумя нулями – 100. Наибольшее двузначное число на 1 меньше наименьшего трёхзначного, то есть 99.

Теперь в цикле *for* мы перебираем все числа заданной длины. Например, для двузначных чисел переменная цикла `num` изменяется от $100 : 10 = 10$ до 99:

```
for (int num = max / 10; num <= max - 1; ++num)
{
    if (!Prime(num))
        continue;
```

Поскольку исходное, длинное число должно быть простым, то и его необходимо проверить, и если оно не соответствует предъявленным требованиям, цикл переходит к проверке *следующего* числа.

Тут нам необходима **функция для проверки заданного числа на простоту**:

```
bool Prime(int number)
{
    //проверяем, делится ли введенное число на числа
    //2..корень квадратный из числа;
    for (int i = 2; i <= sqrt(number); ++i)
    {
        if (number % i == 0)
        {
            return false;
        }
    }
    return true;
}
```

Она возвращает значение *true*, если число *number* простое, и *false* – если составное.

Если длинное число *num* оказалось простым, то нам необходимо проверить и все его составные части. Для этого мы делим его на вспомогательную переменную *i*. При начальном значении, равном 10, мы разобьём число *num* на такие части:

```
1373 / 10 = 137
1373 % 10 = 3
```

В следующей итерации значение переменной *i* станет равным $10 \times 10 = 100$, и мы получим такие части исходного числа:

```
1373 / 100 = 13
1373 % 100 = 73
```

Вы легко продолжите этот процесс расщепления до самого конца. А мы должны убедиться, что каждая из частей также является простым числом. Для этого мы, естественно, опять вызываем функцию *Prime*. Если проверяемое число окажется составным (функция вернёт *false*), то следующие составные части проверять бесполезно, поэтому с помощью оператора *break* мы досрочно выходим из цикла *while*:

```
int i = 10;
while (i < max)
{
    if (!Prime(num / i))
        break;
    if (!Prime(num % i))
        break;
    i *= 10;
}
if (i == max)
    //печатаем суперпростое число:
    cout << num << endl;;
```

}

На выходе (досрочном или «нормальном») мы сравниваем конечное значение переменной i с максимальным - max . Легко понять, что если они совпадут, то все составные части исходного числа оказались простыми, и нам ничего не остаётся, как только напечатать это число в *Консольном окне*.

```
Суперпростые числа - □ ×
Введите число 2..7 > 2
11
13
17
23
31
37
53
71
73
Введите число 2..7 > 3
113
131
137
173
197
311
313
317
373
797
Суперпростые числа - □ ×
Введите число 2..7 > 4
1373
1997
3137
3797
7331
Введите число 2..7 > 5
73331
Введите число 2..7 > 6
739397
```

Суперпростых чисел длины 7 не существует.

Числа Лишрел

Вот очень простой 196-алгоритм:

1. Возьмите любое натуральное число num .
2. Найдите его сумму с обращённым числом $num = num + rev$.
3. Если сумма num – не палиндромическое число, то переходите к п.2.

Операцию 2 называют *Перевернуть и сложить (Reverse-Then-Add)*.

Для большинства чисел это процесс заканчивается очень быстро, но некоторые числа порождают бесконечные циклы (или только очень длинные – это никому не известно). Наименьшее из таких упрямых чисел – **196**. Поэтому и алгоритм, и сама проблема включают это число в своё название.

Такие числа называют **числами Лишрел** (по-английски - *Lychrel numbers*). Это название придумал Уэйд Ван Лэндигэм (Wade VanLandingham), переставив буквы в имени своей подруги Cheryl и добавив лично от себя букву L).

Если число «упрямое», то сумма может быстро переполнить переменную типа *int*, поэтому, чтобы продвинуться дальше, этот тип следует заменить более ёмким типом **unsigned long long**.

В **главной функции** пользователь вводит число для проверки. Если оно меньше 1, то программа закрывается. Если оно изначально палиндромическое (например, 11 или 22), то мы печатаем сообщение об этом, и пользователь возвращается к вводу другого числа:

```
#include <iostream>
#include<windows.h>
```

```

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    system("title Числа Лишрел");

    // бесконечный цикл ввода данных -
    // пока пользователь не закроет программу
    while(true)
    {
        cout << "Введите число > ";
        unsigned long long num;
        cin >> num;
        // если пользователь ввёл число < 1,
        //то программу закрываем:
        if (num < 1)
            return 0;

        cout << endl;

        if (IsPalindromicNumber(num))
        {

            cout << "Число " << num << " уже палиндромическое!";
            cout << endl;
            continue;
        }
    }
}

```

Если «криминала» не обнаружено, программа печатает исходное число и переходит к выполнению описанного выше алгоритма.

В переменной **niter** подсчитывается число шагов, потребовавшихся для получения палиндромического числа (не забывайте, что некоторые числа создают, вероятно, бесконечные последовательности; например число 196 после 2 415 836 итераций превращается в число с 1 миллионом цифр, которое всё ещё не стало палиндромом). По ходу нахождения обращённых чисел и сумм программа печатает их значения:

```

    cout << "Исходное значение num " << num << endl;
    int niter = 0;
    do
    {
        cout << endl;
        unsigned long long rev = ReverseNum(num);
        cout << "Обращённое число num " << rev << endl;
        num += rev;
        cout << "Новое значение num " << num << endl;
        niter += 1;
    }
    while (!IsPalindromicNumber(num));

    cout << "Число итераций = " << niter << endl;
    cout << endl;
}
}

```

Для переворота чисел и проверки их на палиндромичность в программе используются функции **ReverseNum** и **IsPalindromicNumber**:

```

unsigned long long ReverseNum(unsigned long long num)
{
    unsigned long long rev = 0;

    while (num > 0)
    {
        rev = rev*10 + num % 10;
        num /= 10;
    }
    return rev;
}

bool IsPalindromicNumber(unsigned long long num)
{
    unsigned long long rev = 0;
    while (num >= rev)
    {
        rev = 10 * rev + num % 10;
        if (num == rev)

```

```

        return true;

    num /= 10;

    if (num == rev)
        return true;
}
return false;
}

```

Среди чисел, меньших 10000, самым «вредным» оказалось число 89, которое превращается в палиндромическое только после 24 итераций:

```

Числа Лишрел
Введите число > 89
Исходное значение num 89

Обращённое число num 98
Новое значение num 187

Обращённое число num 781
Новое значение num 968

Обращённое число num 869
Новое значение num 1837

Обращённое число num 7381
Новое значение num 9218

Обращённое число num 8129
Новое значение num 17347

Обращённое число num 74371
Новое значение num 91718

Обращённое число num 81719
Новое значение num 173437

Обращённое число num 734371
Новое значение num 907808

Обращённое число num 808709
Новое значение num 1716517

Обращённое число num 7156171
Новое значение num 8872688

Обращённое число num 8862788
Новое значение num 17735476

Обращённое число num 67453771
Новое значение num 85189247

Числа Лишрел
Обращённое число num 74298158
Новое значение num 159487405

Обращённое число num 504784951
Новое значение num 664272356

Обращённое число num 653272466
Новое значение num 1317544822

Обращённое число num 2284457131
Новое значение num 3602001953

Обращённое число num 3591002063
Новое значение num 7193004016

Обращённое число num 6104003917
Новое значение num 13297007933

Обращённое число num 33970079231
Новое значение num 47267087164

Обращённое число num 46178076274
Новое значение num 93445163438

Обращённое число num 83436154439
Новое значение num 176881317877

Обращённое число num 778713188671
Новое значение num 955594506548

Обращённое число num 845605495559
Новое значение num 1801200002107

Обращённое число num 7012000021081
Новое значение num 8813200023188
Число итераций = 24

Введите число >

```


Представленные ниже числа – кандидаты в числа Лишрел:

196, 295, 394, 493, 592, 689, 691, 788, 790, 879, 887, 978, 986, 1495, 1497, 1585, 1587, 1675, 1677, 1765, 1767, 1855, 1857, 1945, 1947, 1997, 2494, 2496, 2584, 2586, 2674, 2676, 2764, 2766, 2854, 2856, 2944, 2946, 2996, 3493, 3495, 3583, 3585, 3673, 3675

Будьте с ними осторожны!

Другие палиндромические диковинки:

Число 10911 становится палиндромическим через 55 шагов:
4668731596684224866951378664

Число 1 186 060 307 891 929 990 после 261 итерации превращается в палиндром, в котором 119 цифр:

44562665878976437622437848976653870388884783662598425855963436955
852489526638748888307835667984873422673467987856626544

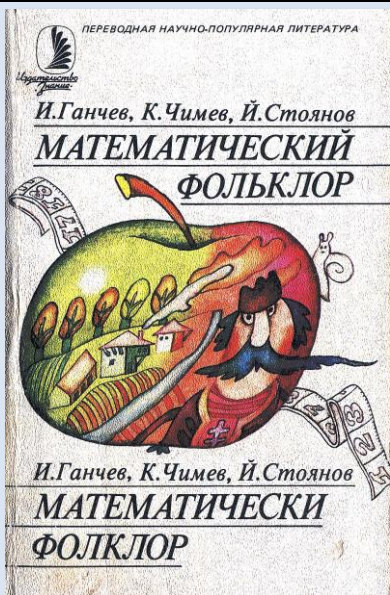
Это число считается мировым рекордсменом. Его нашёл Джейсон Дусетт (Jason Doucette) 30 ноября 2005 года. На его сайте

<http://www.jasondoucette.com/worldrecords.html#Most>

вы найдёте немало интересного о палиндромических числах.

Литература

[Фольклор]

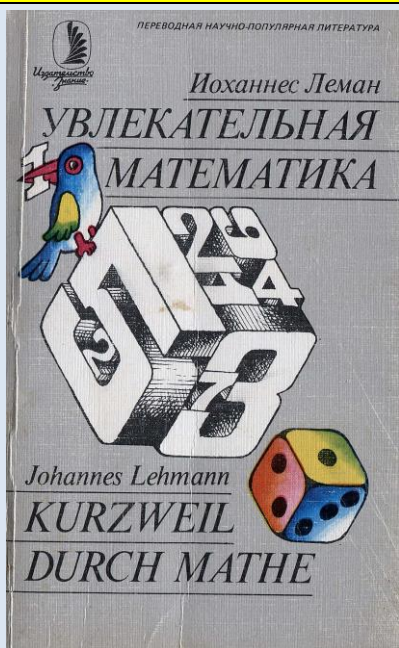


Ганчев, Чимев, Стоянов

Математический фольклор

Знание. Москва, 1985. – 208 с.

[Увлекательная математика]



Иоханнес Леман

Увлекательная математика

Знание. Москва, 1985. – 272 с.

[XP92]



БИБЛИОТЕЧКА «КВАНТ»
выпуск 83

Р. ХОНСБЕРГЕР

МАТЕМАТИЧЕСКИЕ ИЗЮМИНКИ



Росс Хонсбергер

Математические изюминки

Наука, 1992. - 176 с.
Библиотечка «Квант». Вып. 83

ISBN 5-02-014406-1

[ГМ72]



Гарднер Мартин

Математические досуги

М.:Мир, 1972. – 495 с.

Серия Программирование для детей

[Скретч01]

Программирование для детей

Занимательные уроки со Скретчем

Рубанцев Валерий



Рубанцев Валерий

Программирование для детей
Занимательные уроки со *Скретчем*

RVGames, 2016. – 260 с.

Занимательные проекты по различным школьным предметам и компьютерные игры.

[Скретч02]

Программирование для детей

Занимательные уроки со Скретчем

НОВЫЕ



Рубанцев Валерий

Программирование для детей
Новые занимательные уроки со *Скретчем*

RVGames, 2016. – 180 с.

Занимательные проекты по различным школьным предметам и компьютерные игры.

[Скретч03]



Рубанцев Валерий

Программирование для детей
Самые новые интересные уроки
со Скретчем

RVGames, 2016. – 180 с.

Занимательные проекты по различным школьным предметам и компьютерные игры.

[Скретч04]



Рубанцев Валерий

Программирование для детей
Лучшие интересные уроки со
Скретчем

RVGames, 2016. – 160 с.

Занимательные проекты по различным школьным предметам и компьютерные игры.

[Скретч05]

Программирование для детей

Занимательные задачи со Скретчем

Рубанцев Валерий



Рубанцев Валерий

Программирование для детей
Занимательные задачи со *Скретчем*

RVGames, 2016. – 160 с.

Решение занимательных математических задач на *Скретче*.

[Скретч06]

Развивающее программирование

Скретч в первом классе

Рубанцев Валерий



Рубанцев Валерий

Программирование для детей
Скретч в первом классе

RVGames, 2016. – 180 с.

В книге есть всё, чтобы успешно начать программировать на языке программирования *Скретч*.

Интересные проекты, ещё более интересные задачи и головоломки.

[Скретч07]

Развивающее программирование

Скретч во втором классе

Рубанцев Валерий



Рубанцев Валерий

Программирование для детей
Скретч во втором классе

RVGames, 2016. – 220 с.

Продолжаем изучать *Скретч* во втором классе! Программирование, арифметика, алгебра, геометрия.

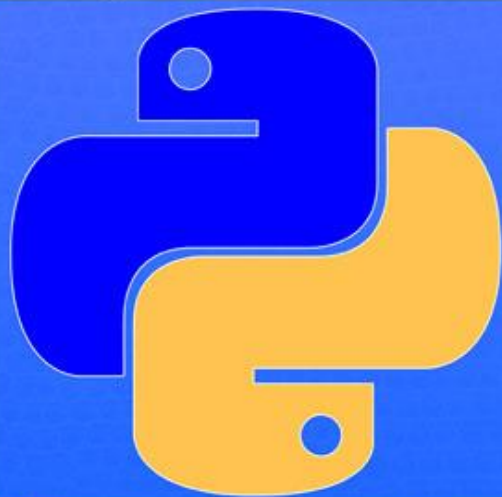
Много проектов - учебных и игровых, компьютерная игра Раскраска, головоломки и интересные сведения о числах

[Питон]

Рубанцев Валерий

Развивающее программирование

Практикум по решению задач на языке Питон 3
Базовый уровень



RVGames



Рубанцев Валерий

Развивающее программирование
Практикум по решению задач на языке *Питон 3*. Базовый уровень

RVGames, 2016. – 500 с.

В книге подробно рассматривается решение более 100 задач: математических, словесных, комбинаторных, вероятностных, игровых.

Лучшие упражнения для отработки навыков программирования на языке *Питон*.

[Си-шарп]

Рубанцев Валерий



КОМПЬЮТЕР, НАУКА И ЖИЗНЬ

ЗАНИМАТЕЛЬНЫЕ МАТЕМАТИЧЕСКИЕ ЗАДАЧИ

От древности
до современности



Рубанцев Валерий

Компьютер, наука и жизнь

Занимательные математические задачи: От древности до современности

RVGames, 2016. – 500 с.

Около 150 проектов на языке *Си-шарп*, показывающих, как можно решать разнообразные занимательные математические задачи на компьютере.

[Геогейбра]

Высокие технологии в школе

Занимательные задачи с Геогейброй

Рубанцев Валерий



Рубанцев Валерий

Высокие технологии в школе

Занимательные задачи с Геогейброй

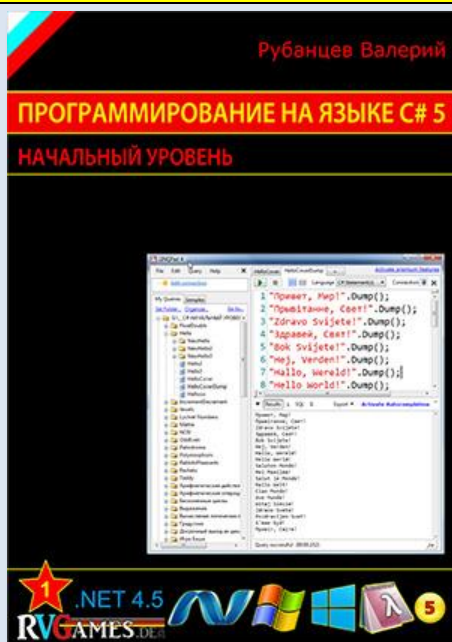
RVGames, 2016. – 160 с.

Решение занимательных математических задач в среде *Геогейбра*.

Несколько десятков интересных задач по всем школьным разделам алгебры.

Серия Программирование на языке C# 5.0: Начальный уровень

[CS10]



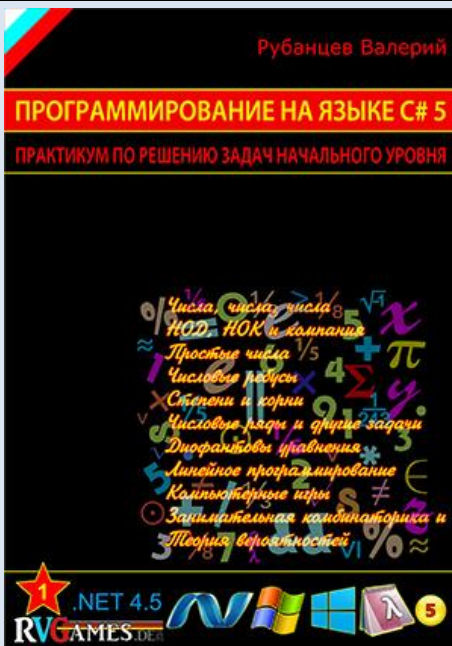
Рубанцев Валерий

Программирование на языке C# 5:
Начальный уровень

RVGames, 2014. – 620 с.

Основы программирования на языке
Си-шарп.

[CS11]



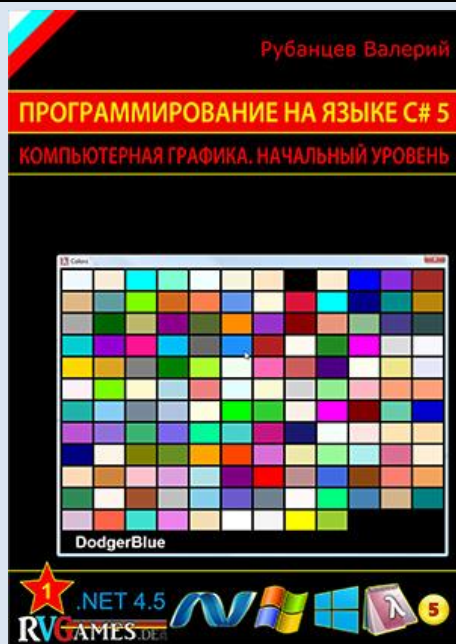
Рубанцев Валерий

Программирование на языке C# 5:
Практикум по решению задач началь-
ного уровня

RVGames, 2014. – 420 с.

Многочисленные проекты для укреп-
ления навыков программирования на
языке *Си-шарп*.

[CS12]



Рубанцев Валерий

Программирование на языке C# 5:
Компьютерная графика. Начальный
уровень

RVGames, 2014. – 460 с.

Основы компьютерной графики.

[CS13]



Рубанцев Валерий

Программирование на языке C# 5:
Тотальный тренинг по *Си-шарпу*.
Начальный уровень

RVGames, 2014. – 400 с.

Разнообразные полезные и занима-
тельные проекты на языке *Си-шарп*.