

Ravesli [Ravesli](#)

- [Уроки по C++](#)
- [OpenGL](#)
- [SFML](#)
- [Qt5](#)
- [RegExр](#)
- [Ассемблер](#)
- [Купить .PDF](#)



Урок №25. Разработка ваших первых программ

[Юрий](#) |

- [Уроки C++](#)

|

Обновл. 2 Сен 2020 |

42081

[↑](#) 29

При написании программ, у вас, как правило, есть какая-то проблема, которую нужно решить. Новички очень часто спотыкаются на этапе преобразования идеи решения проблемы в реальный код. Но самое главное, что вам нужно запомнить — разработка программы выполняется перед этапом написания её кода.

Во многих отношениях, программирование — это как архитектура. Что произойдет, если вы попытаетесь построить дом без соблюдения архитектурного плана? Дом может вы и сумеете построить, но какой он будет: кривые стены, протекающая крыша и т.д. Аналогично, если вы начнете программировать что-нибудь серьезное перед тем, как составите план, то очень скоро обнаружите, что ваш код имеет очень много проблем, на решения которых вы потратите гораздо больше времени/усилий/нервов, нежели на изначальное составление хорошего плана.

Оглавление:

[Шаг №1: Определите проблему](#)

[Шаг №2: Определите свой инструментарий, цели и план бэкапа](#)

[Шаг №3: Разбейте проблему на части](#)

[Шаг №4: Определите последовательность событий](#)

[Шаг №5: Определите данные ввода и данные вывода на каждом этапе](#)

[Шаг №6: Позаботьтесь о деталях](#)

[Шаг №7: Соедините данные ввода с данными вывода в программе](#)

[Советы](#)

Шаг №1: Определите проблему

Первое, что вам нужно сделать — это определить проблему, которую решит ваша программа. В идеале, вы должны сформулировать это одним или двумя предложениями. Например:

- Я хочу написать программу-справочник для удобного хранения и редактирования всех телефонных номеров и звонков.
- Я хочу написать программу для генерации случайных чисел, с помощью которой можно будет определять победителей разных конкурсов.
- Я хочу написать программу, которая будет отслеживать и анализировать акции на фондовом рынке для генерации выигрышных прогнозов.

Хотя этот шаг кажется очевидным, но он также очень важен. Самое худшее, что вы можете сделать — это написать программу, которая делает не то, что вам нужно!

Шаг №2: Определите свой инструментарий, цели и план бэкапа

Для опытных программистов на этом этапе будет еще немало дополнительных пунктов:

- Какая ваша целевая аудитория и какие у нее потребности?
- На какой архитектуре/ОС ваша программа будет работать?
- Какой инструментарий вы будете использовать?
- Будете ли вы разрабатывать программу в одиночку или в составе команды?
- Анализ требований.
- Определение стратегий тестирования/обратной связи/релиза.
- Создание плана [бэкапа](#) в случае неожиданных проблем.

Новички, как правило, большим количеством вопросов не задаются: «Пишу программу для собственного использования, в одиночку, на своей операционной системе, с помощью своей [IDE](#), пользоваться этой программой буду только я». Всё просто.

Если же вы будете работать над чем-нибудь посерьезнее, то стоит еще задуматься о плане бэкапа вашей программы/проекта. Это не просто скопировать код в другую папку ноутбука (хотя это уже лучше, чем ничего). Если ваша ОС «сломается», то вы потеряете все данные. Хорошая стратегия резервного копирования включает в себя создание копии вашего кода вне вашей операционной системы, например:

- Отправить самому себе E-mail с кодом (прикрепить как файл).
- Скопировать в Dropbox или в любое другое облако.
- Перенести на внешнее запоминающее устройство (например, на портативный жёсткий диск).
- Скопировать на другой компьютер в локальной сети.
- Воспользоваться системами контроля версий (например, [GitHub](#), [GitLab](#) или [Bitbucket](#)).

Шаг №3: Разбейте проблему на части

В реальной жизни нам часто приходится выполнять очень сложные задачи. Понять, как их решить, также бывает очень трудно. В таких случаях можно использовать **метод деления на части** (или **метод «от большого к малому»**). То есть, вместо того, чтобы решать одну большую сложную задачу, мы разбиваем её на несколько подзадач, каждую из которых проще решить. Если эти подзадачи все еще слишком сложные, то их также нужно еще раз разбить. И так до тех пор, пока мы не доберемся до точки, где каждая отдельно взятая задача — легко решается.

Рассмотрим это на примере. Допустим, что нам нужно написать доклад о картошке. Наша иерархия задач в настоящее время выглядит следующим образом:

→ Написать доклад о картошке

Это довольно большая задача, давайте разделим её на подзадачи:

■ Написать доклад о картошке

→ Поиск информации о картошке

→ Создание плана

→ Заполнение каждого пункта плана подробной информацией

→ Заключение

Это уже проще, так как теперь мы имеем список подзадач, на которых можем сосредоточиться в индивидуальном порядке. Тем не менее, в данном случае, «Поиск информации о картошке» звучит немного расплывчато, нужно дополнительно разбить и этот пункт:

■ Написать доклад о картошке

■ Поиск информации о картошке

→ Сходить в библиотеку за книжками о картошке

→ Поискать информацию в Интернете

→ Делать заметки в соответствующих разделах из справочного материала

■ Создание плана

→ Информация о выращивании

→ Информация об обработке

→ Информация об удобрениях

■ Заполнение каждого пункта плана подробной информацией

■ Заключение

Выполняя каждый подпункт этого задания, мы решим одну большую задачу.

Есть еще один метод создания иерархии — **метод «от малого к большому»**. Рассмотрим на примере.

Большинство из нас вынуждены ходить на работу (школу/университет) по будням. Предположим, что нам нужно решить проблему «от постели к работе». Если бы вас спросили, что вы делаете перед тем, как добраться на работу, вы бы ответили примерно следующее:

- Выбрать одежду
- Одеться
- Позавтракать
- Ехать на работу
- Почистить зубы
- Встать с постели
- Приготовить завтрак
- Принять душ

Используя метод «от малого к большому», мы можем сгруппировать задания и создать иерархию:

- От постели к работе
 - Спальня
 - Встать с постели
 - Выбрать одежду
 - Одеться
 - Ванная
 - Принять душ
 - Почистить зубы
 - Завтрак
 - Сделать завтрак
 - Позавтракать
 - Транспорт
 - Ехать на работу

Использование подобных иерархий чрезвычайно полезно в программировании для определения структуры всей программы. Задача верхнего уровня (например, «Написать доклад о картошке» или «От постели к работе») становится функцией `main()` (так как это основная проблема, которую нужно решить). Подзадачи становятся функциями в программе.

Шаг №4: Определите последовательность событий

Теперь, когда ваша программа имеет структуру, пришло время ответить на вопрос: «А как же связать все эти пункты вместе?». Первый шаг заключается в определении последовательности событий. Например, когда вы просыпаетесь утром, в какой последовательности вы выполняете вышеперечисленные дела? Скорее всего, в следующей:

- Встать с постели
- Выбрать одежду
- Принять душ
- Одеться
- Приготовить завтрак
- Позавтракать
- Почистить зубы
- Ехать на работу

Если бы мы создавали калькулятор, то последовательность заданий выглядела бы следующим образом:

- Получить первое значение от пользователя
- Получить математическую операцию от пользователя
- Получить второе значение от пользователя
- Вычислить результат
- Вывести результат

Этот список определяет содержимое функции `main()`:

```
1 int main()
2 {
3     getOutOfBed();
4     pickOutClothes();
5     takeAShower();
6     getDressed();
7     prepareBreakfast();
8     eatBreakfast();
9     brushTeeth();
10    driveToWork();
11 }
```

Или, в случае с калькулятором:

```
1 int main()
2 {
3     // Получить первое значение от пользователя
4     getUserInput();
5
6     // Получить математическую операцию от пользователя
```

```
7 |   getMathematicalOperation();
8 |
9 |   // Получить второе значение от пользователя
10 |  getUserInput();
11 |
12 |  // Вычислить результат
13 |  calculateResult();
14 |
15 |  // Вывести результат
16 |  printResult();
17 | }
```

Шаг №5: Определите данные ввода и данные вывода на каждом этапе

После определения иерархии задач и последовательности событий, нам нужно определить, какими будут данные ввода и данные вывода на каждом этапе.

Например, первая функция из вышеприведенного примера — `getUserInput()`, довольно проста. Мы собираемся получить число от пользователя и вернуть его обратно в caller. Таким образом, прототип функции будет выглядеть следующим образом:

```
1 | int getUserInput();
```

В примере с калькулятором, функции `calculateResult()` требуется 3 ввода данных: два числа и 1 математический оператор. При вызове `calculateResult()` у нас уже должны быть 3 фрагмента данных, которые мы будем использовать в качестве параметров функции. Функция `calculateResult()` вычисляет значение результата, но не выводит его. Следовательно, нам необходимо вернуть этот результат в качестве возвращаемого значения обратно в caller, чтобы другие функции также имели возможность его использовать.

Учитывая это, прототип функции становится следующим:

```
1 | int calculateResult(int input1, int op, int input2);
```

Шаг №6: Позаботьтесь о деталях

На этом этапе нам нужно реализовать каждый подпункт из нашей иерархии задач. Если вы разбили задание на достаточно мелкие кусочки, то выполнить каждый из этих кусочков будет несложно. Например:

```
1 | int getMathematicalOperation()
2 | {
3 |     std::cout << "Please enter which operator you want (1 = +, 2 = -, 3 = *, 4 = /)
4 |
5 |     int op;
6 |     std::cin >> op;
7 |
8 |     // А что, если пользователь введет некорректный символ?
9 |     // Пока что мы это проигнорируем
```

```
10 |  
11 |     return op;  
12 | }
```

Шаг №7: Соедините данные ввода с данными вывода в программе

И, наконец, последний шаг — это соединение данных ввода с данными вывода. Например, вы можете отправить выходные данные функции `calculateResult()` во входные данные функции `printResult()`, чтобы вторая функция могла вывести результат. Чаще всего в таких случаях используются промежуточные переменные для временного хранения результата и его перемещения между функциями. Например:

```
1 // Переменная result - это промежуточная переменная, которая используется для переда  
2 int result = calculateResult(input1, op, input2);  
3 printResult(result);
```

Согласитесь, что вариант, приведенный выше, читабельнее варианта без использования временных переменных (см. ниже):

```
1 printResult(calculateResult(input1, op, input2));
```

Рассмотрим готовую версию программы-калькулятора, её структуру и перемещение данных:

```
1 #include <iostream>  
2  
3 int getUserInput()  
4 {  
5     std::cout << "Please enter an integer: ";  
6     int value;  
7     std::cin >> value;  
8     return value;  
9 }  
10  
11 int getMathematicalOperation()  
12 {  
13     std::cout << "Please enter which operator you want (1 = +, 2 = -, 3 = *, 4 = /)  
14  
15     int op;  
16     std::cin >> op;  
17  
18     // А что, если пользователь введет некорректный символ?  
19     // Пока что мы это проигнорируем  
20  
21     return op;  
22 }  
23  
24 int calculateResult(int x, int op, int y)  
25 {  
26     // Обратите внимание, оператор == используется для сравнения двух значений  
27
```

```
28     if (op == 1) // если пользователь выбрал операцию сложения (№1)
29         return x + y; // то выполняем эту строку
30     if (op == 2) // если пользователь выбрал операцию вычитания (№2)
31         return x - y; // то выполняем эту строку
32     if (op == 3) // если пользователь выбрал операцию умножения (№3)
33         return x * y; // то выполняем эту строку
34     if (op == 4) // если пользователь выбрал операцию деления (№4)
35         return x / y; // то выполняем эту строку
36
37     return -1; // вариант, если пользователь ввел некорректный символ
38 }
39
40 void printResult(int result)
41 {
42     std::cout << "Your result is: " << result << std::endl;
43 }
44
45 int main()
46 {
47     // Получаем первое значение от пользователя
48     int input1 = getUserInput();
49
50     // Получаем математическую операцию от пользователя
51     int op = getMathematicalOperation();
52
53     // Получаем второе значение от пользователя
54     int input2 = getUserInput();
55
56     // Вычисляем результат и сохраняем его во временной переменной
57     int result = calculateResult(input1, op, input2 );
58
59     // Выводим результат
60     printResult(result);
61 }
```

Здесь есть несколько концепций, которые мы еще не рассматривали: условное ветвление `if` и использование оператора равенства `==` для сравнения двух элементов. Не беспокойтесь, если вы это не понимаете — мы всё это детально рассмотрим на следующих уроках.

Советы

Пусть ваши первые программы будут простыми. Очень часто новички ставят слишком высокие планки для своих первых более-менее серьезных программ. Например, «Я хочу написать игру с графикой, звуком, монстрами, подземельями и городом, в котором можно будет продавать найденные вещи». Если вы попытаетесь написать что-нибудь подобное в начале вашего пути как программиста, то очень скоро у вас пропадет любое желание программировать. Вместо этого, пусть ваши первые цели/задания/программы будут попроще, например, «Я хочу написать программу, которая отображала бы 2D-поле на экране».

Добавляйте новый функционал со временем. Как только вы написали простенькую программу, которая работает (даже без сбоев), то следующим вашим шагом должно стать добавление нового функционала. Например, когда вы сможете отображать 2D-поле на экране — добавьте персонажа, который сможет ходить по этому полю. После того, как вы уже сможете ходить — добавьте стены, которые будут препятствовать вашему движению. После того, как у вас будут стены — постройте из них город. После того, как у вас будет город — добавьте персонажей-продавцов. При таком подходе на вас не наваливается всё сразу и вы знаете с чего начинать, что делать дальше, в какой последовательности и т.д.

Фокусируйтесь только на одном задании в определенный промежуток времени. Не пытайтесь сделать всё и сразу, не распыляйтесь на несколько задач одновременно. Сосредоточьтесь на одном. Лучше иметь одно выполненное задание и пять невыполненных, нежели шесть частично выполненных заданий. Если вы рассеиваете свое внимание в нескольких направлениях, то и ошибок будет много.

Тестируйте каждую новую часть кода. Начинающие программисты часто пишут программу за один присест. Затем, при компиляции проекта, получают сотни ошибок. Поэтому, после написания определенной части кода — сразу компилируйте и тестируйте её. Если ваш код не будет работать, то вы будете знать, где находится проблема, и исправить её будет намного легче. Как только вы убедились, что ваш код рабочий — переходите к написанию следующей части, а затем *repeat*. Тестирование может занять больше времени, но в конечном итоге ваш код будет работать так, как вам нужно.

Большинство новичков пропускают некоторые из этих шагов и советов, так как это не столь захватывающе, как, собственно, сам процесс кодинга. Хорошая новость заключается в том, что как только вы освоите все эти концепции — они станут для вас естественными в процессе разработки ваших программных продуктов.

Оценить статью:

★★★★★ (578 оценок, среднее: 4,97 из 5)



[← Урок №24. Конфликт имен и std namespace](#)

[Урок №26. Отладка программ: степпинг и точки останова](#)



Комментариев: 29



1.:

[15 августа 2019 в 17:03](#)

Здравствуйтесь, у меня проюлема в коде, ошибок нет, но выводит в результате "-1", почему?

```
1 #include <iostream>
2
3 using namespace std;
```

```
4
5
6 int Usergetcout()
7 {
8     cout << "Enter a number ";
9     int value;
10    cin >> value;
11    return value;
12 }
13 int math1()
14 {
15     cout << "Enter a znachenie(1 = +, 2 = -, 3 = *, 4 = /)" << endl;
16     int op;
17     cin >> op;
18     return op;
19 }
20 int math2(int op, int x, int y)
21 {
22     if (op == 1)
23         return x + y;
24     if (op == 2)
25         return x - y;
26     if (op == 3)
27         return x * y;
28     if (op == 4)
29         return x / y;
30
31     return -1;
32 }
33 void printresult(int result)
34 {
35     cout << "Your result is " << result << endl;
36 }
37 int main()
38 {
39     int input1 = Usergetcout();
40     int op = math1();
41     int input2 = Usergetcout();
42     int result = math2(input1, op, input2);
43     printresult(result);
44 }
```

Ответить



1. *Алексей:*

[17 августа 2019 в 19:11](#)

А вот посмотрите-ка внимательно: вы присваиваете значение переменной result, приравнивая его к результату функции math2 с аргументами input1, op, input2. (То есть `int result = math2(input1, op, input2);`). Но в функции math2 у вас другая последовательность аргументов: `int math2(int op, int x, int y)`. Таким образом в функции math2 значению op будет

присвоено значение `input1`, а оно наверняка больше чем 4, поэтому ваши условия не выполняются, и возвращается -1.

[Ответить](#)

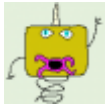


2. *Алексей:*

[7 августа 2019 в 02:41](#)

Юрий, в вашей программе — калькуляторе при вводе пользователем неверного числа или символа — на экран выводится "-1", что некорректно, и непонятно пользователю. Нужно добавить дополнительное условие для проверки корректности введенных данных сразу в первой функции, раз уж начали применять условия. При вводе пользователем неверного символа сразу выводить на экран "Fucken shit!" или "Введенное значение некорректно", и делать возврат в функцию ввода. Так будет удобнее и понятнее пользователю, и избавит от ошибок в типах данных. Тогда пользователю не удастся случайно "сломать" программу, и это тоже показатель.

[Ответить](#)



3. *Михаил:*

[25 июня 2019 в 14:01](#)

```
1  #include <iostream>
2  #include <Windows.h>
3  #include <cmath>
4
5  using namespace std;
6
7  //вычисления корня
8
9
10
11 //Первая функция принятия данных от пользователя
12
13 int dannieOtPolzovatel()
14 {
15     std::cout << "Введите цифру одной из сторон" << endl;
16     int dannie;
17     std::cin >> dannie;
18     return dannie;
19 }
20
21 //Вторая функция получение 2 данных
22
23 int dannieOtPolzovatel2()
24 {
25     std::cout << "Введите цифру втоой из сторон" << endl;
26     int dannie2;
27     std::cin >> dannie2;
28     return dannie2;
29 }
```

```

30
31 //Вычисление диагонали по формуле пифагора  $a^2+b^2=c^2$ 
32 double reshenie(double a, double b)
33 {
34     double c = NULL;
35     c = (a * a)+( b * b);
36     double ax = sqrt(c);
37     return ax;
38 }
39
40
41 void printResult(double result)
42 {
43     std::cout << "Результат диагональ равна " << result << std::endl;
44 }
45
46 int main()
47 {
48
49     SetConsoleCP(1251);
50     SetConsoleOutputCP(1251);
51     std::cout << "Моя первая программа вычисляет дигональ треугольника" << std
52
53     int in = dannieOtPolzovatel();
54     int in2 = dannieOtPolzovatel2();
55     double result = reshenie(in,in2);
56     printResult(result);
57
58
59     return 0;
60 }
61 /*А тут у меня мой первый многострочный комментарий
62 1
63 2
64 3*/

```

[Ответить](#)



1. **КАРЫЧ:**

[20 июля 2019 в 21:51](#)

ЭМ... А зачем использовать namespace std и в итоге писать std::

[Ответить](#)



2. **Константин:**

[8 мая 2020 в 23:43](#)

Михаил, хотел тебе подсказать, мол, зачем писать две функции для получения данных от пользователя, ну и всё такое, но наткнулся на диагональ в треугольнике — сейчас сию, грузусь...

[Ответить](#)

4. Анастасия:

[22 июня 2019 в 17:06](#)

Попыталась объединить все знания, полученные за 25 уроков, и вот что вышло. (всё работает, но хотелось бы получить рекомендации на счёт практичности и релевантности, которые были бы более выгодны в сложных проектах).

Основной файл с `main()`, в котором вызываются функции `MyFirstProgramm.cpp`

```
1 #include "pch.h"
2 #include <iostream>
3 #include "Functions.h"
4
5 using namespace std;
6 int main()
7 {
8     double digit1 = firstStep();           //вызываем функцию для
9     char action = Operator();              //эта функция для ввода
10    double digit2 = thirdStep();           //функция для взятия
11    double finish = Determination(digit1, action, digit2); //работаем над числами
12    Output(finish);                        //выводим результат
13    return 0;
14 }
```

Файл с задачей на взятие первого значения `firstStep.cpp`

```
1 #include "pch.h"
2 #include <iostream>
3
4 double firstStep()
5 {
6     std::cout << "Hello, my friend! " << "Enter any number, please >> ";
7     double digit1;
8     std::cin >> digit1;
9     return digit1;
10 }
```

Файл с задачей на взятие оператора `Operator.cpp`

```
1 #include "pch.h"
2 #include <iostream>
3
4 char Operator()
5 {
6     std::cout << "Further: input some mathematical operation ->> ";
7     char action;
8     std::cin >> action;
9     return action;
10 }
```

Снова на взятие значения, но тут уже другой текст "просьбы", так сказать, для разнообразия (но можно было бы снова вызвать функцию firstStep()) thirdStep.cpp

```
1 #include "pch.h"
2 #include <iostream>
3
4 double thirdStep()
5 {
6     std::cout << "Eventually, enter another one number >> ";
7     double digit2;
8     std::cin >> digit2;
9     return digit2;
10 }
```

Теперь самое важное — произвести вычисления. Можно было бы и с помощью цикла, но выбор пал именно на switch. Determination.cpp

```
1 #include "pch.h"
2 #include <iostream>
3
4
5 double Determination(double digit1, char action, double digit2)
6 {
7     std::cout << "          Compute..." << std::endl;
8
9     switch (action) {
10    case '+':
11        return digit1 + digit2;
12        break;
13    case '-':
14        return digit1 - digit2;
15        break;
16    case '*':
17        return digit1 * digit2;
18        break;
19    case '/':
20        if (digit2 == 0){
21            std::cout << "\n          To zero cannot be split!";
22            return 0;
23        }
24        else
25            return digit1 / digit2;
26        break;
27    default:
28        std::cout << "\nSomething went wrong. \nTry again, please.";
29    }
30 }
```

И сам вывод. Output.cpp (Да, не лучшее название...)

```
1 #include "pch.h"
2 #include <iostream>
3
```

```

4 double Output(double finish) {
5     if (finish != 0) {
6         std::cout << "          Been waiting long?\n" << "          And that ar
7         return 0;
8     }
9     else
10    return 0;
11 }

```

А так же файл заголовка Functions.h . Работаю на VS поэтому обошлась #pragma once

```

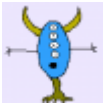
1 #pragma once
2 double firstStep();
3 char Operator();
4 double thirdStep();
5 double Determination(double, char, double);
6 double Output(double);

```

На счёт странных пропусков:

Хотелось как-то упорядочить вывод строк, но ничего другого не пришло в голову, кроме как оставить пробелы в нужных местах.

Ответить



5. zashiki:

6 июня 2019 в 15:18

Хотела объединить с уроком "многофайловые программы", все работает. Но есть какие то замечания?

Заголовочный файл, объявляющий функции: opstates.h:

```

1 #ifndef DBL
2 #define DBL
3 int getNum(); //объявление функции "получить число"
4 int getOperNum(); //объявление функции "получить оператор"
5 int getResult(int a, int b, int c); //объявление функции "возвратить результат"
6 #endif

```

файл для получения чисел: get1n.cpp:

```

1 #include<iostream>
2 //функция "получить число"
3 int getNum()
4 {
5     int a;
6     std::cin >> a;
7     return a;
8 }

```

файл для получения знаков: get2o.cpp:

```
1 #include<iostream>
2 //функция получить оператор для чисел
3 int getOperNum()
4 {
5     int a;
6     std::cout << "\nКакую операцию хотите совершить с числами?: ";
7     std::cout << "\nесли +, то введите 1";
8     std::cout << "\nесли -, то введите 2";
9     std::cout << "\nесли *, то введите 3";
10    std::cout << "\nесли /, то введите 4\n\n";
11    std::cin >> a;
12
13    return a;
14 }
```

файл для возвращения результата: get3r.cpp:

```
1 #include<iostream>
2 //функция вернуть результат
3 int getResult(int a, int b, int c)
4 {
5     std::cout << "\n";
6
7     if (c == 1)
8     {
9         std::cout << "ваш результат: " << a << " + " << b << " = ";
10        return a + b;
11    }
12    else if (c == 2)
13    {
14        std::cout << "ваш результат: " << a << " - " << b << " = ";
15        return a - b;
16    }
17    else if (c == 3)
18    {
19        std::cout << "ваш результат: " << a << " * " << b << " = ";
20        return a * b;
21    }
22    else
23    {
24        std::cout << "ваш результат: " << a << " / " << b << " = ";
25        return a / b;
26    }
27
28
29 }
```

файл main.cpp:

```
1 #include<iostream>
2 #include "opstates.h"
3
```



```

4  int main()
5  {
6      setlocale(0, ""); //не выходит с windows.h, даже с изменением шрифта
7
8      /*начало калькулятора:*/
9
10     std::cout << "Введите 1е число: ";
11     int a = getNum(); //инициируется 1я переменная
12
13     std::cout << "Введите 2е число: ";
14     int b = getNum(); //инициируются 2я переменная
15
16     std::cout << getResult(a, b, getOperNum()); //выводится результат с 2мя пер
17
18     /*конец калькулятора*/
19
20     std::cin.clear();
21     std::cin.ignore(32767, '\n');
22     std::cin.get();
23     return 0;
24 }

```

[Ответить](#)



6. **Сергей:**

[5 июня 2019 в 20:13](#)

Тест. Как вставить программный код чтобы он был отформатирован как в примерах?

```

1  #include <iostream>
2
3  int main(int result)
4  {
5      cout << "Hello, world!";
6      return 0;
7  }

```

Отформатируется ли текст автоматически?

[Ответить](#)



1. **Юрий:**

[6 июня 2019 в 14:50](#)

Ребята, уже не первый раз говорю. Отправляйте код, как есть — всё форматирование выполняется на моей стороне с помощью определённого плагина. От вас никакого форматирования текста/кода не требуется — только copy/paste. Остальное я сделаю сам.

[Ответить](#)

7. *Михайло:*[25 марта 2019 в 20:55](#)

Но как в программе калькулятор одни функции знают о переменных с других функций у них же локальная область видимости.

[Ответить](#)1. *Cerberus:*[27 марта 2019 в 05:56](#)

Функции знают ровно то, что им передано в качестве аргументов. Где-то этот принцип нарушается, на Ваш взгляд?

[Ответить](#)8. *Денис:*[13 февраля 2019 в 01:57](#)

Я немного иначе, по простому сделал, может и не правильно, но работает. Из следующих уроков ничего нет, только базовые знания этого курса:) Спасибо за доходчивую статью:)

```
1 #include "stdafx.h" // only MS Visual Studio
2 #include <iostream>
3
4 using namespace std;
5
6 int getUserInput() //Запрос числа у пользователя
7 {
8     int x;
9     cout << "Введите число: ";
10    cin >> x;
11    return x;
12 }
13
14 int operation() //Запрос операции с числами
15 {
16     int op;
17     cout << "Введите действие (1 = +; 2 = -; 3 = *; 4 = /): ";
18     cin >> op;
19     return op;
20 }
21
22 int calcresultat(int x, int op, int y) //Основная функция, которая высчитывает
23 {
24     if (op == 1)
25         return x + y;
26     if (op == 2)
27         return x - y;
28     if (op == 3)
```

```

29     return x * y;
30     if (op == 4)
31         return x / y;
32     else // оператор if - можно читать как "если", else - "иначе" (для новичков)
33         cout << "Выбранное действие отсутствует!" << endl;
34
35
36     return 0;
37 }
38
39
40 int main()
41 {
42     setlocale(0, ""); // подключаем русский (может что-то ещё, я не проверял :)
43     int a = getUserInput(); // первое число
44     int o = operation(); //действие
45     int b = getUserInput(); // второе число
46     int c = calcresultat(a, o, b); // отправляем значения в функцию
47     cout << "Результат: " << c << endl; // выводим результат
48     system("pause"); // приходится ставить паузу, ибо консоль по завершению цит
49
50     return 0;
51 }

```

Если где-то тупо или допустил ошибку, я открыт для критики, это даже полезно на данном этапе:) Только очень прошу, критикуйте доходчиво:)

Ответить



1. Nikita:

[11 мая 2019 в 23:04](#)

А где собственно изменения?

Ответить



9. Вячеслав:

[2 января 2019 в 20:40](#)

вот посмотрите, что скажете?

```

1  #include <stdio.h>
2  #include <iostream>
3
4  int main(){
5      int a = 0;
6      int b = 0;
7      char operation;
8      std::cout << "Enter first number: ";
9      std::cin >> a;
10     std::cout << "Enter second number: ";

```

```
11     std::cin >> b;
12     std::cout << "Enter operation: ";
13     std::cin >> operation;
14
15     switch (operation){
16     case '-':
17         std::cout << "a - b = " << a - b << '\n';
18         break;
19     case '+':
20         std::cout << "a + b = " << a + b << '\n';
21         break;
22     case '*':
23         std::cout << "a * b = " << a * b << '\n';
24         break;
25     case '/':
26         std::cout << "a / b = " << a / b << '\n';
27         break;
28     default:
29         std::cout << "Error\n";
30     }
31     return 0;
32 }
```

[Ответить](#)



1. *Alexey:*

[25 января 2019 в 02:15](#)

```
1  #include "pch.h"
2  #include <iostream>
3  #include <Windows.h>
4
5  char count = 'a';
6
7  int main() {
8      SetConsoleCP(1251);
9      SetConsoleOutputCP(1251);
10     int a = getUserInput();
11     int b = getUserInput();
12     int result = calc(a, b);
13     printResult(result);
14     return 0;
15 }
16
17 int getUserInput() {
18     std::cout << "Введите число " << count++ << std::endl;
19     int inputValue = 0;
20     std::cin >> inputValue;
21     return inputValue;
22 }
23
```

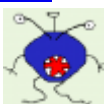
```
24 int getOperators() {
25     std::cout << "Выберите необходимое действие: \n 1) + \n 2) - \n 3) * \n 4) / \n 0) выход: ";
26     int op = 0;
27     std::cin >> op;
28     return op;
29 }
30
31 int calc(int a, int b) {
32     Label1:
33     int op = getOperators();
34     switch (op) {
35     case 1:
36         return a + b;
37         break;
38     case 2:
39         return a - b;
40         break;
41     case 3:
42         return a * b;
43         break;
44     case 4:
45         if (b != 0) {
46             return a / b;
47         }
48         else {
49             std::cout << "На 0 делить нельзя!" << std::endl;
50             return 0;
51         }
52         break;
53     default:
54         std::cout << "Вы ввели недоступное значение, попробуйте еще раз."
55         goto Label1;
56         break;
57     }
58 }
59
60 void printResult(int result) {
61     std::cout << "Ваш результат: " << result << std::endl;
62 }
```

[Ответить](#)

10. **Вячеслав:**

[2 января 2019 в 18:44](#)

у меня проблем нет с калькулятором все работает как швейцарские часы

[Ответить](#)

1. **ThatSameGuy:**

[26 января 2019 в 23:02](#)

Прямо как часы ?

i.imgur.com/5erIMgw.png

(Фикситься очевидно превращением а и b в float)

[Ответить](#)



11. Виктор:

[4 декабря 2018 в 13:45](#)

Здравствуйте. При написании калькулятора я решил подключить кириллицу. Прописал так как вы писали. Можете объяснить что означают эти строки:

```
1 SetConsoleCP(1251);
2 SetConsoleOutputCP(1251);
```

И, еще я чет не могу догнать когда нужно использовать int, а когда void. Можете как-то популярнее мне объяснить? И, посмотрите на мои комментарии строк, правильно ли я все понимаю. Заранее спасибо за ответ.

```
1 #include "stdafx.h"
2 #include <iostream> //подключение библиотеки которая содержит функциональные в
3 #include <Windows.h> //заголовочный файл который позволяет использовать кириллицу
4
5
6 using namespace std; //объявление пространства имен std
7 int getUserInput() //функция получения пользовательского ввода
8 {
9     int x; //объявление целочисленной переменной x
10    cout << "Введите число: "; //вывод в консоль
11    cin >> x; //присваивание переменной x значение пользовательского ввода
12    return x; //возвращение переменной x в caller
13
14 }
15 int getMathematicalOperation() //функция получения математической операции
16 {
17     int op; //объявление целочисленной переменной op
18    cout << "Выберите оператор: 1=+, 2=-, 3=*, 4=: "; //вывод в консоль
19    cin >> op; //присваивание переменной op значение пользовательского ввода
20    return op; //возвращение переменной op в caller
21
22 }
23 int calculateResult(int x, int op, int y) //функция вычисления результата
24 {
25     if (op == 1) //если пользователь введет 1 то
26         cout << x << "+" << y; //вывод в консоль
27     if (op == 1) //если пользователь введет 1 то
28         return x + y; //возвращение значения x + y в caller
29
30     if (op == 2)
31         cout << x << "-" << y;
32     if (op == 2)
33         return x - y;
```

```

34
35     if (op == 3)
36         cout << x << "*" << y;
37     if (op == 3)
38         return x * y;
39
40     if (op == 4)
41         cout << x << "/" << y;
42     if (op == 4)
43         return x / y;
44
45     return -1; // стандартное значение "error", если пользователь введет
46 }
47 void printResult(int result) //функция вывода результата в консоль
48 {
49     cout << "=" << result<< endl; //вывод результата в консоль
50 }
51 int main(int result)
52 {
53     SetConsoleCP(1251); //????
54     SetConsoleOutputCP(1251); //????
55     int input1 = getUserInput(); //объявление переменной input и присва
56     int op = getMathematicalOperacion(); //объявление переменной op и пр
57     int input2 = getUserInput(); ////объявление переменной input2 и при
58     result = calculateResult(input1, op, input2); //присваивание переменн
59     printResult(result); //вызов функции с одним аргументом printResult
60     return 0; //возвращение значения 0 в caller
}

```

Ответить



1. Константин:
[23 декабря 2018 в 00:34](#)

Виктор, каждое второе if выбросить, а взамен блок:

```

1 if(op == 1)
2 {
3     cout << x << " + " << y ;
4     return x + y;
5 }

```

и тип int поставь перед переменной result, т.е. д.б.

```

1 int main()
2 {
3     //...
4     int result = calculateResult(input1, op , input2);
5     //...
6     return 0;
7 }

```

а вообще и так всё работает, и понятие есть А 1251 это, видимо, какой-то параметр, который включает кириллицу во встроенных функциях ввода и вывода в — , на — консоль.

[Ответить](#)



2. Константин:

[23 декабря 2018 в 00:42](#)

Даже не так: 1251 — что-то радикальнее, чем просто ввод-вывод кириллицы на экран, делает — меняет кодировку проекта (хотя деталей этого я не знаю)

[Ответить](#)



12. Наиль:

[2 сентября 2018 в 14:38](#)

Написал примерно такой же калькулятор. первое число принимается, операция принимается, второе число принимается, и дальше ничего ответ не выдаётся что делать?

[Ответить](#)



13. Дима:

[6 июня 2018 в 11:56](#)

Супер) Все как в жизни

[Ответить](#)



14. rainkiller:

[20 мая 2018 в 18:33](#)

Хорошая статья. Интуитивно со всем согласен.

Было некое уныние от непонимания концепций заголовочных файлов и header guards, но, к счастью, преодолел себя.

[Ответить](#)



1. Юрий:

[20 мая 2018 в 19:10](#)

Спасибо, главное — не останавливаться и продолжать дальше.

[Ответить](#)



15. Семён:

[19 марта 2018 в 18:12](#)

Спасибо , огромное !


Конечно , кое что не понятно , НО пока . С таким объяснением , а точнее перевод , понять не будет сложно , всё постепенно !

[Ответить](#)

1.  Юрий:
[19 марта 2018 в 19:16](#)

Пожалуйста 😊

[Ответить](#)

16.  Иван:
[3 марта 2018 в 22:27](#)

Красава, сразу видно что сам наступал на грабли и решил нас предостеречь, от лишних хлопот. мало того что умеешь объяснить, так ещё и убедить! Не возникает желание сделать принципиально не так! Спасибо автор.

[Ответить](#)

1.  Юрий:
[3 марта 2018 в 22:47](#)

Автор не я, но со всем этим согласен 😊

[Ответить](#)

Добавить комментарий

Ваш E-mail не будет опубликован. Обязательные поля помечены *

Имя *

Email *

Комментарий

☐ Сохранить моё Имя и E-mail. Видеть комментарии, отправленные на модерацию






☐ Получать уведомления о новых комментариях по электронной почте. Вы можете [подписаться](#) без комментирования.

[TELEGRAM](#)  [КАНАЛ](#)



ПАБЛИК 

ТОП СТАТЬИ

-  [Словарь программиста. Сленг, который должен знать каждый кодер](#)
-  [Урок №1. Введение в программирование](#)
-  [70+ бесплатных ресурсов для изучения программирования](#)
-  [Урок №1: Введение в создание игры «Same Game»](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

- [Ravesli](#)
- - [О проекте](#) -
- - [Пользовательское Соглашение](#) -
- - [Все статьи](#) -
- Copyright © 2015 - 2020