По одной из классификаций, имеющей значение в программировании, алгоритмы делятся на следующие:

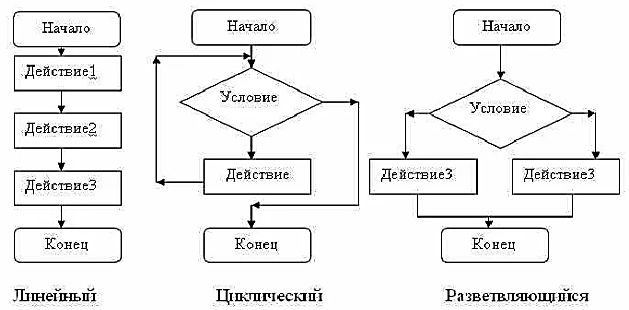
******

Рис 1. Виды алгоритмов

Рассмотрим более подробно линейный алгоритм.

***Линейный*** алгоритм – набор команд (указаний), выполняемых последовательно во времени друг за другом. Рассмотрим пример:



Рис 2. Линейный алгоритм

На практике, подобные алгоритмы часто реализуются при решении задач вычислительного характера. Например, для реализации действий, заданных формулой типа площади круга

Но и для решения подобных задач не всегда можно обойтись без проверки условий.

Рассмотрим несколько простых задач вычислительного характера.

**Вычисление площади круга**

Введем переменные:

s – вещественное число для площади круга. После расчета выдается на экран.

pi – вещественное число для записи значения 3.14

r – вещественное число для записи радиуса. Радиус будем вводит с клавиатуры.

Для записи алгоритма сначала используем псевдокод.

**АЛГ** ПлКр

pi=3.14

**ВВОД** r

s=pi\*r\*r

**ВЫВОД** s

**КОНЕЦ** ПлКР

При вычислении s использовался знак \*. Это знак умножения. В большинстве языков аналогично, как и в С++. Кроме этого используются и другие арифметические операции:

+ сложение

- вычитание

\* умножение

/ деление

Легко заметить, что возведения в степень отсутствует, т.к. в языках программирования обычно лишнего нет. Всякие дополнительные возможности обычно выносятся во внешние библиотеки. Такие библиотеки требуется «заказать», после чего дополнительные возможности становятся доступными.

Реализуем вычисление на С++

**#include "stdafx.h" //необходима для некоторых версий С++**

**#include <iostream> //библиотека для ввода-вывода**

**using namespace std; //пространство имен std**

**void main() //название программы без типа**

**{ //начало программы**

**float pi = 3.14;**

**float s, r;**

**cin >> r; //ввод с клавиатуры в переменную r**

**s = pi\*r\*r; //вычисление площади**

**cout << s; //выдача на экран переменной s**

**} //конец программы**

Если выполнить эту программу, то она выполнит расчет и выдаст результат на экран, но его мы не успеем рассмотреть. Поэтому после выдачи **s** сделаем паузу. Для этого часто используется специальный системный оператор system. Далее приводится только тело программы:

**void main()**

**{**

**float pi = 3.14;**

**float s, r;**

**cin >> r;**

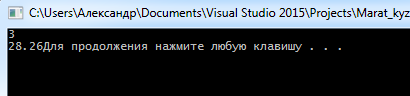
**s = pi\*r\*r;**

**cout << s;**

**}**

После запуска программы произойдет остановка на операторе cin, где программа будет ждать ввода радиуса. Для того что бы программа продолжила работу, пользователь должен ввести любое значение радиуса и нажать клавишу Enter.

В нашем примере радиус равен 3. Получим такой результат:



Очень некрасиво. Программа должна перед вводом сообщить, что требуется ввести, а также оформить выдачу результата. Немного переделаем программу:

**void main()**

**{**

**float pi = 3.14;**

**float s, r;**

**cout << "Vvedite radius - ";**

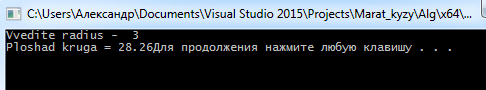
**cin >> r;**

**s = pi\*r\*r;**

**cout << "Ploshad kruga = " << s;**

**}**

Сообщения написаны английскими буквами. Можно использовать и кириллицу, но для этого потребуются задавать кодовую таблицу. Пока отложим этот решение этого вопроса. Результат будет таким:



Сообщение “Для продолжения нажмите любую клавишу ...” сливается с результатом. Для красоты добавим последний штрих:

**void main()**

**{**

**float pi = 3.14;**

**float s, r;**

**cout << "Vvedite radius - ";**

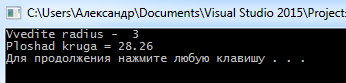
**cin >> r;**

**s = pi\*r\*r;**

**cout << "Ploshad kruga = " << s << endl;**

**}**

Здесь употребили константу **endl** (системная константа «переход на следующую строку»). Часто вместо endl используют **'\n'**



Данный вариант уже приемлемый.

**Операции с переменными разного типа.**

Предположим, что имеются целые и дробные числа и требуется их сложить. Для человека эта операция представляется тривиальной. Например, 5+3.12=8.12.

Попробуем эту операцию выполнить в С++, для чего введем два числа x и y, сложим и запишем в переменную s, которую выдадим на экран.

Код программы:

**#include "stdafx.h"**

**#include <iostream>**

**using namespace std;**

**void main()**

**{**

**float x,y;**

**int s;**

**cout << "x = ";**

**cin >> x;**

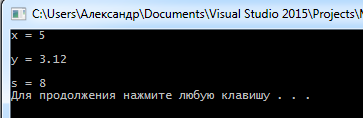
**cout <<endl << "y = ";**

**cin >> y;**

**s = x+y;**

**cout << endl << "s = " << s << endl;**

**}**



Программа сложила два числа 5 и 3.12, но результат неправильный. Почему? Это произошло потому, что переменная s объявлена целой, а переменные x и y –вещественные. Структура хранения целых и вещественных разные, поэтому компьютер просто отбрасывает дробную часть. Не округляет, а отбрасывает.

Такой же результат мы получим в следующем случае:

**void main()**

**{**

**int x,y;**

**float s;**

**cout << "x = ";**

**cin >> x;**

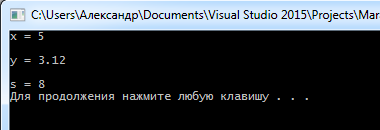
**cout <<endl << "y = ";**

**cin >> y;**

**s = x+y;**

**cout << endl << "s = " << s << endl;**

**}**

****

Теперь сделаем правильно:

**void main()**

**{**

**float x,y;**

**float s;**

**cout << "x = ";**

**cin >> x;**

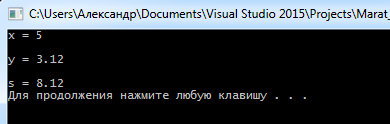
**cout <<endl << "y = ";**

**cin >> y;**

**s = x+y;**

**cout << endl << "s = " << s << endl;**

**}**



Этот пример показывает, как важно следить за типами переменных. Такую ошибку в сложных расчетах очень трудно найти.

Разберем еще один пример в котором участвует переменная логического типа (bool).

Выполним сложение вещественного числа и с переменной логического типа. Например, 3.12+true

По житейской логике такое не имеет смысла. Для многих языков программирования — это также запрещено (выдается сообщение об ошибке). Но для С++ такое сложение допускается. Каков же будет результат?

**#include "stdafx.h"**

**#include <iostream>**

**using namespace std;**

**void main()**

**{**

**float x;**

**bool y;**

**float s;**

**cout << "x = ";**

**cin >> x;**

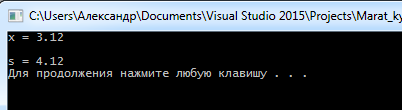
**y=true;**

**s = x+y;**

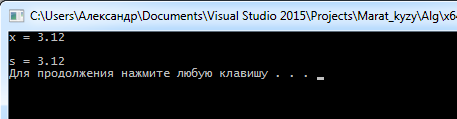
**cout << endl << "s = " << s << endl;**

**}**

Результат работы:



Легко заметить что значение true в С++ воспринимается как единица. Возможно что false – это ноль. Присвоим переменной y=false и проверим:



Видим, что false читается как ноль.

Видоизменим немного текст программы:

**#include <iostream>**

**using namespace std;**

**void main()**

**{**

**float x;**

**bool y;**

**float s;**

**cout << "x = ";**

**cin >> x;**

**cout << "y = ";**

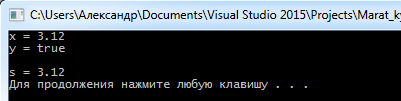
**cin >> y;**

**s = x + y;**

**cout << endl << "s = " << s << endl;**

**}**

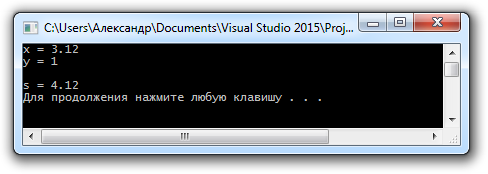
В этом варианте попытаемся ввести логическое выражение true в переменную y.

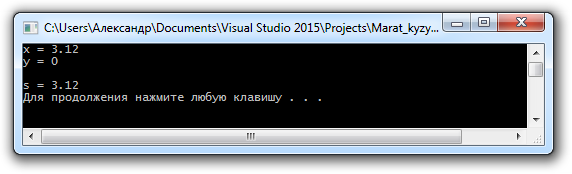


Убеждаемся, что ввести можно, но это значение просто игнорируется. Причем сообщение об ошибке не выдается.

Возможно это недоработка, которая может приводить к ошибкам. Программисту следует учитывать такие недоработки. Они конечно есть и в других языках.

Как же выполнить ввод в переменную y, которая имеет логический тип. Вспомним что такое логический тип. Это два значения: ноль –false и единица – true. Попробуем такой вариант:





Наше предположение верное.

**Разберем решение еще одной задачи: «загадать число».**

1. Программа просит загадать целое число от 1 до 100, но не вводить его.
2. Мысленно прибавить к нему 10
3. Умножить результат на 2
4. Вычесть 5
5. Прибавить задуманное
6. Ввести с клавиатуры полученный результат?
7. Выдать на экран задуманное.

(Х + 10)\*2 – 5 +Х = SUM, где SUM – сколько получилось

2\*Х +20 -5 +Х = SUM

3\*X= SUM -15

Х=( SUM -15)/3

**Программа на С++**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int x, sum;**

**cout << "Загадайте число (1 : 9)?" << endl;**

**cin.get(); //пауза. Для продолжения нажать Enter**

**cout << "Прибавьте 10" << endl;**

**cin.get();**

**cout << "Результат умножить на 2" << endl;**

**cin.get();**

**cout << "Вычесть - 5" << endl;**

**cin.get();**

**cout << "К результату прибавьте задуманное число" << endl;**

**cin.get();**

**cout << "Какое число получилось? Введите - ";**

**cin >> sum;**

**x = (sum - 15) / 3;**

**cout << "Вы задумали число = " << x << endl;**

**cin.get();**

**}**

Как видно из программы – задуманное число не вводится, а рассчитывается на основе полученной пользователем суммы.

Проверим работу программы. Загадаем число 7.

