**Занятие 1. Базовые знания о языке программирования С++**

**Программа** – это реализация алгоритма для выполнения задачи компьютером (ЭВМ).

С помощью программы мы формулируем алгоритм на языке, понятном компьютеру. Таким языком служит язык программирования.

На языке С++ можно составлять программы для инженерных расчетов, также можно строить оконные проекты, имеющие пользовательский графический интерфейс.

Существует несколько приложений для написания программ на языке С++. В языке программирования С++ определены некоторые стандартные типы данных, которые представлены в [табл. 3.1](https://www.intuit.ru/studies/courses/16740/1301/lecture/25626?page=1#table.3.1).

|  |  |
| --- | --- |
| Таблица 3.1. Стандартные типы данных С++ | |
| **Тип данных** | **Значение** |
| int | целый тип, размер типа int не определяется стандартом, а зависит от компьютера и компилятора, для 16-разрядного процессора под величины этого типа отводится 2 байта, для 32-разрядного — 4 байта.Примеры значений типа int: 5, 0, -1, 100. |
| double | вещественный тип с двойной точностью. Типы данных с плавающей точкой хранятся в памяти компьютера иначе, чем целочисленные. Внутреннее представление вещественного числа состоит из двух частей — мантиссы и порядка. Мантисса — это число, большее 1.0, но меньшее 10. Для величин типа double, занимающих 8 байт, под порядок и мантиссу отводится 11 и 52 разряда соответственно. Длина мантиссы определяет точность числа, а длина порядка — его диапазон.Примеры значений типа double: 5.0, -0.00001, 2.9987. |
| float | вещественный тип. В компьютерах величины типа float занимают 4 байта, из которых один двоичный разряд отводится под знак мантиссы, 8 разрядов под порядок и 23 под мантиссу. Тип float имеет меньшую точность, чем double. В большинстве случаев лучше использовать double. |
| char | символьный тип, под величину символьного типа отводится количество байт, достаточное для размещения любого символа из набора символов для данного компьютера, что и обусловило название типа. Как правило, это 1 байт. Каждый символ имеет свой собственный целочисленный код, согласно таблице ASCII (англ. American Standard Code for Information Interchange).Примеры значений типа char: "A" (код 65), "7" (код 55), "-" (код 189), "/" (код 191). |
| fstream, ifstream, ofstream | файловый поток. Используется для вывода данных в файл и считывания данных из файла. |

Программа на языке С++ имеет определенную структуру.

Существует определенная последовательность заранее определенных строк кода, которая приведена в [табл. 3.2](https://www.intuit.ru/studies/courses/16740/1301/lecture/25626?page=1#table.3.2).

|  |  |
| --- | --- |
| Таблица 3.2. Структура программы С++ | |
| #include "stdafx.h" | подключение заголовочного файла для сборки проекта. Обязательный пункт в Visual Studio, в среде Borland C++ не используется. |
| #include <название\_библиотеки> | подключение библиотек. Необязательный пункт. Подробно о библиотеках смотреть ниже. |
| using namespace std; | использование пространства имен. Обязательный пункт в Visual Studio, в среде Borland C++ (это специализированное програмное обеспечение, которое представляет собой консольный компилятор. )не используется. |
| int main(){ или void main(){ | главная функция программы. Именно она начинает выполняться, когда запускается программа. Обязательный пункт. |
| Тело\_функции\_main | в теле функции main записываются действия и операции, предусмотренные алгоритмом. Обязательный пункт. |
| return 0;} или } | конец программы. Самый последний оператор. Обязательный пункт. |

**Библиотека** – это файл, в котором описаны функции и операторы. Для каждой смысловой группы функций своя библиотека. Библиотек в языке C++ предусмотрено много, мы будем рассматривать только самые необходимые. Нам понадобятся функции для работы с экраном, математические функции.

Основные библиотеки и их основные функции и операторы:

1. <iostream> для Visual Studio, <iostream.h> для Borland C++. Библиотека для работы с консолью (экраном).

*Cin - это объект входного потока пространства имен std: std::cin >> x;*

*cin, операция взятия из потока >> чтобы получить от пользователя введенное им значение. Объект std::cin забирает вводимую пользователем информацию из стандартного потока ввода, который обычно является клавиатура.*

cout – оператор вывода данных на экран.

Пример использования:

cout<<"fraza"; //выведет на экран слово fraza. Может вывести любой текст.

cout<<x; //выведет на экран число, хранящееся в переменной х.

cin – оператор считывания с экрана.

Когда у пользователя запрашивают число, программа ждет, пока пользователь не напечатает число и нажмет ENTER. Тогда оператор cin записывает это значение в переменную х.

Пример использования:

cin>>x; //присваивает переменной х значение, введенное с клавиатуры.

cin>>x>>y; //присваивает переменной х первое введенное с клавиатуры значение, переменной у – второе.

endl – оператор перевода каретки на экране на следующую строку.

Самостоятельно не используется.

Пример использования:

cout<<endl; //курсор перейдет на новую строку.

cout<<x<<endl;//сначала на экране появится число, хранящееся в переменной х, потом перейдет на новую строку. Выводимые далее данные будут печататься с новой строки.

cout<<endl<<"fraza"; // курсор перейдет на новую строку, и на новой строке появится надпись fraza.

precision(n) – функция для отображения на экране дробных чисел с n цифрами после запятой (точность).

Пример использования:

cout.precision(3)<<7.897426; //число 7.897426 выведется на экран в виде 7.897.

1. <math.h> одинаково для Visual Studio, Borland C++. Библиотека математических функций. Основные математические функции представлены в [табл. 3.3](https://www.intuit.ru/studies/courses/16740/1301/lecture/25626?page=1#table.3.3).

|  |  |  |
| --- | --- | --- |
| Таблица 3.3. Основные математические функции С++ | | |
| **Математическая функция** | **Программная запись** | **Описание** |
| |x| | fabs(x) | Модуль числа. |
| \sin x | sin(x) | Синус числа, аргумент в радианах. |
| \cos x | cos(x) | Косинус числа, аргумент в радианах. |
| \tg x | tan(x) | Тангенс числа, аргумент в радианах. |
| e^x | exp(x) | Экспонента числа. |
| \ln x | log(x) | Натуральный логарифм числа. |
| \lg x | log10(x) | Десятичный логарифм числа. |
| x^y | pow(x, y) | х в степени y. |
| 10^x | pow10(x) | Степень десяти. |
| \sqrt{x} | sqrt(x) | Квадратный корень из числа. |
| \arcsin x | asin(x) | Арксинус числа, в радианах. |
| \arccos x | acos(x) | Арккосинус числа, в радианах. |
| \arctg x | atan(x) | Арктангенс числа, в радианах. |
| \pi | M\_PI | Число \pi=3.141593 |

1. <iomanip> для Visual Studio, <iomanip.h> для Borland C++.

setw(n) – для вывода на экран отводится n ячеек.

Используется при построении ровной таблицы значений функции.

Пример использования:

cout<<setw(5)<<x<<setw(5)<<y<<endl;

На экране выведутся два числа: первое (1.5) в первых пяти ячейках, второе (-73) во вторых пяти ячейках. \_ \_ 1 . 5 \_ \_ - 7 3 .

В программах используются переменные. Имя переменной выбирает составитель программы; имя переменной должно начинаться с буквы латинского алфавита и может содержать буквы латинского алфавита, цифры и символы подчеркивания. Заглавные и строчные буквы считаются разными. Примеры имен переменных: х, y, summa, s1, srednee\_ar и т.д. Имена переменных не должны совпадать с ключевыми словами языка С++.

Чтобы использовать в программе переменную, необходимо:

1. объявить переменную в начале программы, явно указав тип данных для переменной. Пример:

double x; //вещественная переменная х.

int m; //целочисленная переменная m.

Если переменная не будет объявлена, но будет использоваться далее в программе, то программа не запустится, компилятор выдаст ошибку.

1. проинициализировать переменную, т.е. задать переменной значение. Пример:
2. x=7.81;

m=4; z=x+m;

Если переменная не будет проинициализирована, то компилятор не выдаст ошибки, но расчеты будут выполнены неверно.

1. использовать далее в программе в расчетах или при выводе на экран.

Для числовых переменных определены простейшие арифметические операции, которые приведены в [табл. 3.4](https://www.intuit.ru/studies/courses/16740/1301/lecture/25626?page=1#table.3.4). Для их использования не нужно подключать библиотеку.

|  |  |  |
| --- | --- | --- |
| Таблица 3.4. Основные арифметические операции С++ | | |
| **Операция в С++** | **Описание** | **Пример в программе** |
| = | присвоение | х=0.4; |
| + | сложение | х=8+3;  y=x+7.9; |
| - | вычитание | x=8-3;  y=x-7.9; |
| \* | умножение | x=4\*2.5;  y=x\*4; |
| / | деление. Результатом деления целых чисел является целое число. | x=7.5/3.2;  y=x/2.6;  z=1/3; //z=0 (округление до целого, т.к. делятся целые числа)  z=1.0/3.0; //z=0.333(3) |

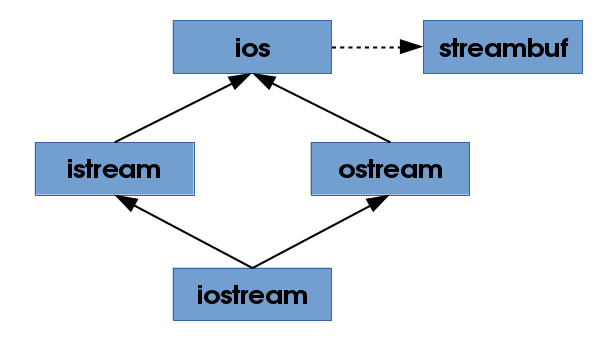
Для сравнения переменных и чисел используются встроенные знаки сравнения, которые приведены в [табл. 3.5](https://www.intuit.ru/studies/courses/16740/1301/lecture/25626?page=1#table.3.5). Для их использования не нужно подключать библиотеку.

|  |  |  |
| --- | --- | --- |
| Таблица 3.5. Логические операции С++ | | |
| **Сравнение в С++** | **Описание** | **Пример в программе** |
| > | больше чем | x>0 |
| < | меньше чем | y<z |
| >= | больше или равно | y>=x |
| <= | меньше или равно | z<=8.56 |
| == | проверка на равенство | x==0.7 |
| != | не равно | x!=y |
| && | логическое И | x>0 && x<1 // двойное неравенство 0<x<1 |
| || | логическое ИЛИ | s<8 || s>10 |

**Библиотека ввода/вывода. Манипуляторы. Флаги форматирования**

* [Потоки](http://inf-w.ru/?page_id=3214#str)
* [Потоковые операции](http://inf-w.ru/?page_id=3214#op)
* [Функции неформатированного ввода/вывода](http://inf-w.ru/?page_id=3214#nonf)
* [Манипуляторы](http://inf-w.ru/?page_id=3214#man)
* [Функции и флаги форматирования](http://inf-w.ru/?page_id=3214#flag)

Потоки

В языке C++ нет встроенной поддержки ввода/вывода, вместо этого используется библиотека функций IOStream. Эта библиотека содержит различные классы, определяющие *потоки* ввода/вывода. ***Класс***, наряду с объектом, является ключевым понятием объектно-ориентированного программирования. ***Объект*** находится в таком же отношении к своему классу, в каком переменная находится по отношению к своему типу. Старшим классом в иерархии является класс ios\_base. Он содержит функции ввода/вывода общие для всех остальных, производных, классов, а также указатель на класс streambuf, значением которого является адрес буфера памяти.  
**Буфер** – это блок памяти, используемый в качестве временного хранилища данных. Буферизация позволяет считывать информацию с устройства большими блоками с размещением в буфере, а затем, самой программе, считывать и записывать информацию в буфер посимвольно.  
[](http://inf-w.ru/wp-content/uploads/2015/07/3.png)  
**Поток** (stream) – это абстракция, отражающая перемещение данных от источника к приемнику. Реализация потоков в языке C++ выполнена таким образом, что потоки для различных источников и приемников выглядят единообразно.

* **Ввод** интерпретируется как чтение данных в поток (с клавиатуры).

Эти потоки определяет класс istream (input stream).

* **Вывод** интерпретируется как запись данных из потока (на дисплей).

Эти потоки определяет класс ostream (output stream).

Наследником классов istream и ostream является класс iostream, следовательно им наследуются и объекты чтения и объекты записи. Для потоков ввода/вывода в библиотеке IOStream определены глобальные объекты чтения и записи:

* объект cin читает данные со стандартного устройства ввода (клавиатура);
* объект cout записывает данные на стандартное устройство вывода (монитор).

Для того, чтобы потоковые объекты стали доступными в программе необходимо использовать директиву компилятора include с именем заголовочного файла, заключенного в угловые скобки <>:  
#include <iostream>  
Имена объектов принадлежат пространству имен стандартной библиотеки. Для более лаконичного обращения к глобальным объектам необходимо сделать объявление с помощью следующей инструкции using:  
using namespace std;

Потоковые операции

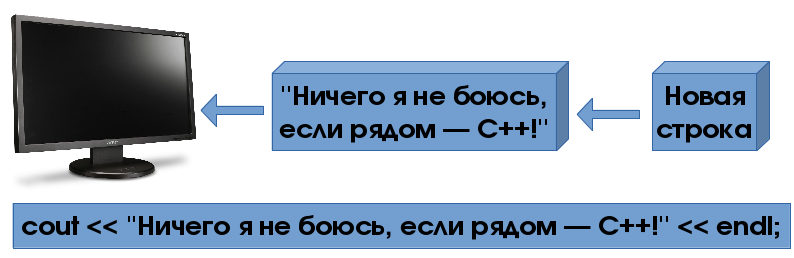
* Для вывода данных используется **операция вывода** или **операция вставки** – operator<<
* Для ввода данных используется **операция ввода** или **операция извлечения** - operator>>

В программе и на схеме это выглядит следующим образом:

cin >> var\_A;

[](http://inf-w.ru/wp-content/uploads/2015/07/1.png)

cout << "Ничего я не боюсь, если рядом - С++!" << endl;

[](http://inf-w.ru/wp-content/uploads/2015/07/2.png)

Здесь var\_A - это переменная, именованная область памяти компьютера.  
Особенностью использования этих операций в том, что они позволяют добавлять в поток данные - "цепочкой" (как в примере, выше, добавляется манипулятор endl), по очередности, так как операции << и >> обладают ассоциативностью слева-направо.  
Пример, как это можно реализовать, в программе:

**Программа ext\_3.1**

**#include <iostream>**

**#include<locale>**

**using namespace std;**

**int main() {**

**setlocale(0,"");**

**int a, b, c;**

**cout<<"Введите 3 числа"<<endl;**

**cin >> a >> b >> c;**

**cout << "Первое число "**

**<< a**

**<< "\nВторое число "**

**<< b**

**<< "\nТретье число "**

**<< c**

**<< "\na + b + c = "**

**<< a + b + c**

**<< "\na \* b \* c = "**

**<< a \* b \* c**

**<< endl;**

**return 0;**

**}**

2 3 4

Первое число 2

Второе число 3

Третье число 4

a + b + c = 9

a \* b \* c = 24

В случае, если инструкция потока, оформленная в виде сцепления нескольких операций >> или <<, окажется слишком длинной, то инструкцию потока можно представить в виде столбца. Такой подход делает программный код более читабельным.

Или так

cout << "Первое число "<< a<< "\nВторое число "<< b<< "\nТретье число "<< c<< "\na + b + c = "<< a + b + c<< "\na \* b \* c = "<< a \* b \* c << endl;

return 0;

}

**Функции неформатированного ввода/вывода**

Классы istream и ostream включают ряд методов, которые в совокупности называются функциями неформатированного ввода/вывода. Некоторые из них представлены в таблице ниже. Эти функции названы так потому, что они производят считывание или запись символьных данных без каких-либо преобразований, не пропуская при этом пробельные символы.

|  |  |
| --- | --- |
| **istream** | **ostream** |
| get() - извлекает символ | put() - записывает символ |
| peek() - читает символ без извлечения |  |
| unget() - возврат извлеченного ранее символа в поток |  |
| putback() - вставка символа в поток |  |
| getline() - извлекает символы, пока не будет найден заданный символ |  |
| ignore() - удаляет из потока определенное количество символов пока не будет встречен символ разделитель, который также отбрасывается |  |
| read() - извлекает из потока блоки символов в целевой массив | write() - вставляет блоки символов в поток |
| gcount() - количество извлеченных из потока символов. Все выше перечисленные функции будут воздействовать на gcount, включая >> (операцию извлечения) |  |

Поскольку эти методы являются функциями классов, то вызов этих методов должен сопровождаться предварительным указанием имени класса, членом которого данная функция является. Имя класса и имя метода этого класса разделяются операцией "точка":

ch = cin.peek();

cin.read(s, 100);

cin.ignore(255, '\n');

cout.put(line);

cout.write(str1, 50);

**Манипуляторы**

В программе ext\_3.1 в потоке вывода используется манипулятор endl. Этот манипулятор вставляет в буфер вывода символ новой строки и выгружает содержимое буфера на стандартное устройство.  
**Манипуляторы** - вспомогательные функции управляющие форматированием потоков ввода/вывода. Помимо endl, еще несколько манипуляторов доступны в заголовке iostream:

* scientific  Экспоненциальная форма вывода вещественных чисел
* fixed       Фиксированная форма вывода вещественных чисел
* dec         Перевод в десятичную систему счисления
* oct         Перевод в восьмеричную систему счисления
* hex         Перевод в шестнадцатеричную систему счисления
* ends        Вставка символа конца строки
* showbase    Выводить индикатор основания систем счисления
* left        Выравнивать по левому краю поля
* boolalpha   Вывод логических величин в тестовом виде
* uppercase   Выводить в числах прописные буквы
* showpos     Выводить знак + для положительных чисел

**Программа ext\_3.2**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int a = 15;**

**cout << "a\_10 = " << a << endl;**

**cout << oct << "a\_\_8 = " << a << endl;**

**cout << hex << "a\_16 = " << a << endl;**

**return 0;**

**}**

a\_10 = 15

a\_\_8 = 17

a\_16 = f

Следующие манипуляторы имеют параметры:

* setw()         Устанавливает ширину поля вывода
* setfill()      Устанавливает символ заполнения (по умолчанию, пробел)
* setprecision() Число выводимых знаков дробной части

Для использования этих манипуляторов необходимо включить заголовок iomanip.

**Программа ext\_3.3**

**#include <iostream>**

**#include <iomanip>**

**using namespace std;**

**int main() {**

**int a = 10;**

**float b = 10.0 / 2.3;**

**cout << a << endl;**

**cout << setw(12) << a << endl;**

**cout << scientific << b << endl;**

**cout << "b = " << fixed << b << endl;**

**cout << setw(12) << setprecision(2) << b << endl;**

**cout << setw(12) << setfill('.') << b << endl;**

**return 0;**

**}**

10

10

4.347826e+00

b = 4.347826

4.35

........4.35

В программах рассмотрены примеры, когда манипуляторы используются в потоках вывода. Однако манипуляторы можно применять и в потоках ввода. Пример использования манипулятора setw для контроля количества вводимых символов:

**Программа ext\_3.4**

**#include <iostream>**

**#include <iomanip>**

**using namespace std;**

**int main() {**

**const int buf = 10;**

**char c[buf];**

**// будет прочитано 9 символов + '\0'**

**cin >> setw(buf) >> c;**

**cout << c << endl;**

**return 0;**

**}**

qwertyuiopasdfghjkl

qwertyuio

**Функции и флаги форматирования**

Для управления вводом/выводом, помимо манипуляторов, доступны флаги форматирования (как аргументы) и форматирующие функции. Эти функции вызываются как методы базового класса (ios) dot-нотацией. Например: cout.width(10).  
Некоторые функции класса ios:

* fill(ch)      Устанавливает символ заполнения
* precision(p)  Устанавливает точность
* width(w)      Устанавливает ширину поля
* setf(flags)   Устанавливает флаг форматирования
* unsetf(flags) Сбрасывает указанный флаг форматирования

**Флаги форматирования** определяют параметры ввода/вывода которые будут действовать во всех операциях ввода/вывода, пока не будут отменены.  
Для включения параметра используется следующий синтаксис:

cout.setf(ios::flag);

cout.setf(ios::flag1 | ios::flag2 | ios::flag3);

Для выключения:

cout.unsetf(ios::flag)

Здесь flag, flag1, flag2... подключаемые или отключаемые флаги. Имена флагов, в большинстве своем, совпадают с именами манипуляторов. В программе можно одновременно использовать и манипуляторы, и флаги форматирования и функции-члены.  
Функция setf может иметь два аргумента. В таком случае, первый аргумент - устанавливаемый флаг, а второй - маска для сброса действующего флага:

* basefield   для dec, oct и hex
* adjustfield для left, right и internal
* floatfield  для scientific и fixed

Например: cout.setf(ios::scientific, ios::floatfield);

**Программа ext\_3.5**

**#include <iostream>**

**#include <iomanip>**

**#include <cmath>**

**using namespace std;**

**int main() {**

**cout.setf(ios::fixed | ios::showpos);**

**cout.fill('.');**

**cout.precision(3);**

**for (int i = 4; i < 20; i++) {**

**cout << i \* sqrt(7);**

**cout.width(15);**

**cout << i \* sqrt(5) << endl;**

**}**

**return 0;**

**}**

+10.583.........+8.944

+13.229........+11.180

+15.875........+13.416

+18.520........+15.652

+21.166........+17.889

+23.812........+20.125

+26.458........+22.361

+29.103........+24.597

+31.749........+26.833

+34.395........+29.069

+37.041........+31.305

+39.686........+33.541

+42.332........+35.777

+44.978........+38.013

+47.624........+40.249

+50.269........+42.485

С полным списком манипуляторов и флагов форматирования вы можете познакомиться в [методичке](http://inf-w.ru/wp-content/uploads/2018/08/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BA%D0%B02.pdf) (Табл. 12 и 13).

Самостоятельная работа

1. Составьте программу печатающую таблицу зависимости значения величины гипотенузы от значений данных катетов по теореме Пифагора в следующем формате:
2. -------------------------
3. | N | A | B | C |
4. -------------------------
5. | 1 | 10 | 15 | 18.028 |
6. | 2 | 20 | 30 | 36.056 |
7. | 3 | 30 | 45 | 54.083 |
8. | 4 | 40 | 60 | 72.111 |
9. | 5 | 50 | 75 | 90.139 |
10. | 6 | 60 | 90 | 108.17 |
11. | 7 | 70 | 105 | 126.19 |
12. | 8 | 80 | 120 | 144.22 |
13. | 9 | 90 | 135 | 162.25 |

-------------------------