C++

Единственный способ изучать новый язык программирования - писать на нем программы.

Брайэн Керниган

История языка Си++

Язык Си++ был разработан в начале 1980-х гг. Бьерном Страуструпом из компании AT&T Bell Laboratories. Си++ основан на языке Си. Два символа "++« в названии – это игра слов, символами "++" в языке Си обозначается операция инкремента (увеличение значения переменной на 1).

История языка Си++

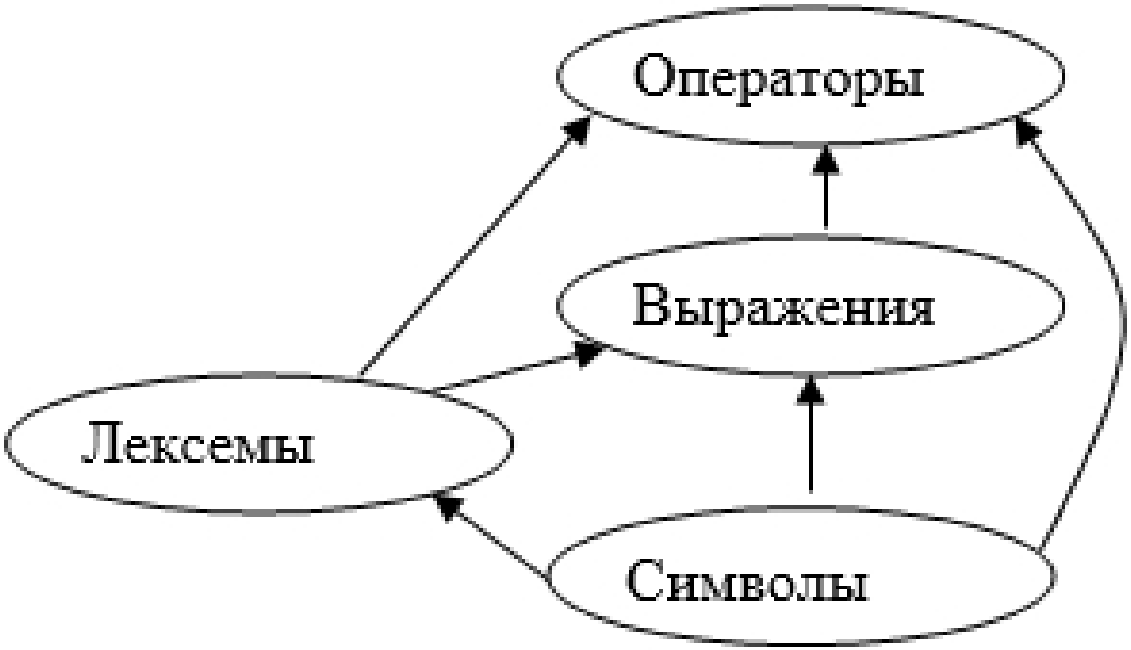
Таким образом, Си++ был задуман как язык Си с расширенными возможностями. Большая часть языка Си вошла в Си++ как подмножество, поэтому многие программы на Си можно скомпилировать (т.е. превратить в набор низкоуровневых команд, которые компьютер может непосредственно выполнять) с помощью компилятора Си++.

Состав языка

- В тексте на любом естественном языке можно выделить четыре основных элемента: символы, слова, словосочетания и предложения.
- Подобные элементы содержит и алгоритмический язык, только слова называют лексемами (элементарными конструкциями), словосочетания — выражениями, а предложения — операторами.

Состав языка

Лексемы образуются из символов, выражения — из лексем и символов, а операторы — из символов, выражений и лексем.



Состав языка

- Операторы бывают исполняемые и неисполняемые.
- *Исполняемые операторы* задают действия над данными.
- *Неисполняемые операторы* служат для описания данных, поэтому их часто называют операторами описания или просто описаниями.

Состав языка

- Принято исходный код программ на C++ сохранять с расширением .cpp после имени файла (происходит такая идея от названия «C Plus Plus» и от того, что во многих операционных системах знак плюс нельзя использовать в именах файлов и каталогов).

Состав языка

Для C++ существует масса IDE (Интегрированная среда разработки, англ. Integrated Development Environment) под Windows

- NetBeans https://netbeans.org/
- DevC++ http://orwelldevcpp.blogspot.com/
- Falcon http://falconcpp.sourceforge.net/
- Microsoft Visual Studio https://www.microsoft.com/
- Eclipse https://www.eclipse.org/
- CodeLite https://codelite.org/

On-line компилятор https://ideone.com/

Состав языка

Внутри IDE (интегрированной среды разработки) процесс компиляции и запуска автоматизирован и, как правило, скрыт от разработчика. Но эти процессы всё равно, происходят каждый раз при попытке запустить программу, притом строго в рассмотренной нами последовательности, т.е. самое минимальное изменение в программном коде требует пересохранения файла с исходным кодом, перекомпиляции и перезапуска программы.

Состав языка

- Перед тем, как создаётся исполняемый код, программа анализируется отладчиком, который ищет в исходном коде существующие и потенциальные ошибки. Если ошибок не найдено, то команда make ничего не выведет на экран в результате своей работы, иначе — будет представлена информация об ошибках с указанием строк, в которых они присутствуют. Пока ошибки не будут исправлены, исполнимый файл не будет создан (или не обновлён, если существовал ранее).

Алфавит языка

Алфавит C++ включает:

1. прописные и строчные латинские буквы и знак подчеркивания;
2. арабские цифры от 0 до 9;
3. специальные знаки, например, {, %, # и т.д.
4. пробельные символы: пробел, символы табуляции, символы перехода на новую строку.

Алфавит языка

Из символов алфавита формируются *лексемы* языка:

- идентификаторы;
- ключевые (зарезервированные) слова;
- знаки операций;
- константы;
- разделители (скобки, точка, запятая, пробельные символы).

Алфавит языка

В тесте программы можно использовать **комментарии**.

Если текст с двух символов «косая черта» // и заканчивается символом перехода на новую строку или заключен между символами /* и */, то компилятор его игнорирует.

Для хранения данных в C++ используются различные сущности, наиболее простыми из них являются литералы, константы и переменные.

Литералом называется явно указанное в исходном коде программы значение определенного типа.

```
cout << 1024; // выводим на экран целочисленный литерал
cout << "mir"; // выводим на экран строковый литерал
```

Идентификаторы

- **Переменной** называется именованная область памяти компьютера (имя которой задаёт разработчик) в которую можно записывать (в том числе повторно, замещая ранее хранимое значение) значения определенного типа и откуда эти значения можно читать.
- При создании любой переменной требуется указать её тип и задать имя.

Идентификаторы

Например:

```
int per1; // создали переменную типа int с именем per1
per1 = 25; // сохранили в переменную целое число 25
int b; // создали переменную типа int с именем b
b = 3 + per1; // прочитали значение 25, сложили его с 3 и сумму записали в b
cout << b; // прочитали из переменной b значение 28 и вывели его на экран
```

Идентификаторы

Использованный в примере тип **int** позволяет хранить целочисленные значения из некоторого диапазона (диапазон будет представлен в таблице далее). Перед тем как использовать переменную (т.е. записывать в неё значение или читать из ней значение) её обязательно нужно объявить (указав её тип и задав имя). Изменить тип переменной или повторно создать переменную — невозможно.

Идентификаторы

Идентификатор — это имя программного объекта.

В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания.

Прописные и строчные буквы различаются.

Первым символом идентификатора может быть буква или знак подчеркивания.

Идентификаторы

Длина идентификатора по стандарту не ограничена. Идентификатор создается на этапе объявления переменной, функции, типа и т.п., после этого его можно использовать в последующих операторах программы.

Константы

Константной называется именованная область памяти, в которую при создании можно записать значение определенного типа, но далее по ходу программы это значение можно только читать (и нельзя изменять).

```
const int k1 = 13; // создали константу типа int с именем k1 и записали в неё значение
cout << k1 = 12; // но нельзя изменить, это приведёт к ошибке
```

Идентификаторы

При выборе идентификатора необходимо иметь в виду следующее:

1. идентификатор не должен совпадать с ключевыми словами и именами используемых стандартных объектов языка;
2. не рекомендуется начинать идентификаторы с символа подчеркивания;
3. на идентификаторы, используемые для определения внешних переменных, налагаются ограничения компоновщика.
4. Для улучшения читаемости программы следует давать объектам осмысленные имена.

Концепция типа данных

Тип данных определяет:

1. *внутреннее представление* данных в памяти компьютера;
2. *множество значений*, которые могут принимать величины этого типа;
3. *операции и функции*, которые можно применять к величинам этого типа.

Все типы языка C++ можно разделить на *простые* (скалярные), *составные* (агрегатные) и *функциональные*. Простые типы могут быть стандартными и определенными программистом.

Концепция типа данных

В языке C++ определено шесть стандартных простых типов данных для представления целых, вещественных, символьных и логических величин.

На основе этих типов, а также массивов и указателей (указатель не является самостоятельным типом, он всегда связан с каким-либо другим конкретным типом), программист может вводить описание собственных простых или структурированных типов.

К структурированным типам относятся перечисления, функции, структуры, объединения и классы.

Простые типы данных

Простые типы делятся на целочисленные типы и типы с плавающей точкой.

Для описания *стандартных типов* определены следующие ключевые слова:

1. int (целый);
2. char (символьный);
3. wchar_t (расширенный символьный);
4. bool (логический);
5. float (вещественный);
6. double (вещественный с двойной точностью).

Простые типы данных

Существует четыре *спецификатора типа*, уточняющих внутреннее представление и диапазон значений стандартных типов:

1. short (короткий);
2. long (длинный);
3. signed (со знаком);
4. unsigned (без знака).

Простые типы данных

Диапазоны значений простых типов данных для IBM PC

Тип	Диапазон значений	Размер (байт)
`bool`	`true` и `false`	1
`signed char`	–128 .. 127	1
`unsigned char`	0 .. 255	1
`signed short int`	–32 768 .. 32 767	2
`unsigned short int`	0 .. 65 535	2
`signed long int`	–2 147 483 648 .. 2 147 483 647	4
`unsigned long int`	0 .. 4 294 967 295	4
`float`	$3.4e^{-38}$ .. $3.4e^{+38}$	4
`double`	$1.7\ e^{-308}$ .. $1.7\ e^{+308}$	8
`long double`	$3.4\ e^{-4932}$ .. $3.4\ e^{+4932}$	10

Простые типы данных

Символьный тип

Данные типа **char** в памяти компьютера всегда занимают 1 байт. Символьный тип может быть со знаком или без него. В величинах со знаком *signed char* можно хранить значение от -128 до 127. Соответственно значения переменных типа *unsigned char* могут находиться в диапазоне от 0 до 255.

При работе с символьными данными нужно помнить, что если в выражении встречается одиночный символ, то он должен быть заключен в одинарные кавычки (‘a’).

Простые типы данных

Целочисленный тип

Переменная типа **int** в памяти компьютера может занимать либо 2, либо 4 байта. Это зависит разрядности процессора. По умолчанию все целые типы считаются знаковыми, то есть спецификатор *signed* можно не указывать. Спецификатор *unsigned* позволяет представлять только положительные числа.

Простые типы данных

Вещественный тип

Число с плавающей точкой представлено в форме mE +- p, где m — мантисса (целое или дробное число с десятичной точкой), p — порядок (целое число). Обычно величины типа float занимают 4 байта, а double 8 байт.

Таблица диапазонов значений вещественного типа:

float	3,4E-38…3,4E+38	4 байта
double	1,7E-308…1,7E+308	8 байт
long double	3,4E-4932…3,4E+4932	8 байт

Простые типы данных

Логический тип

Переменная типа **bool** может принимать только два значения *true (*истина*)* или *fasle (*ложь*).* Любое значение, не равное нулю, интерпретируется как *true.* Значение *false* представлено в памяти как 0.

Простые типы данных

Тип void

Тип void используется для определения функций, которые не возвращают значения, для указания пустого списка аргументов функции, как базовый тип для указателей и в операции приведения типов.

Выражения

Из констант, переменных, разделителей и знаков операций можно конструировать выражения.

- ✓ Каждое выражение представляет собой правило вычисления нового значения.
- ✓ Если выражение формирует целое или вещественное число, то оно называется арифметическим.
- ✓ Пара арифметических выражений, объединенная операцией сравнения, называется отношением.
- ✓ Если отношение имеет ненулевое значение, то оно – истинно, иначе – ложно.

Выражения

Приоритеты операций в выражениях

Ранг	Операции
1	( ) [ ] -> .
2	! ~ — ++ — & * (тип) sizeof тип( )
3	* / % (мультипликативные бинарные)
	+ — (аддитивные бинарные)
5	<< >> (поразрядного сдвига)
6	< > <= >= (отношения)
7	== != (отношения)
8	& (поразрядная конъюнкция «И»)
9	^ (поразрядное исключающее «ИЛИ»)
10	| (поразрядная дизъюнкция «ИЛИ»)
11	&& (конъюнкция «И»)
12	|| (дизъюнкция «ИЛИ»)
13	?: (условная операция)
14	= *= /= %= -= &= ^= |= <<= >>= (операция присваивания)
15	, (операция запятая)

Основные библиотеки

Библиотеки - гардеробы, из которых умелые люди могут извлекать кое-что для украшения, многое - для любопытства и еще больше для употребления.

Дж. Дайер

Библиотека iostream

В первой строке программы с помощью директивы **#include** происходит подключение заголовочного файла **iostream**.

Заголовочные файлы содержат описание функций и других готовых элементов, которые можно использовать в своих программах после того, как заголовочный файл подключён. iostream входит в стандартную библиотеку C++, но заголовочные файлы можно создавать и самостоятельно, помещая туда часто используемые функции, шаблоны и прочие заготовки.

Библиотека iostream

Заголовочный файл **iostream** содержит набор готовых функций для потокового ввода и вывода.

Библиотека iostream

Ввод данных

Ввод данных в C++ осуществляется с помощью команды **cin** (**C**onsole **In**put).

Аргумент этой функции передаётся не в круглых скобках, а через оператор >> (аналог перенаправления в GNU/Linux).

Вывод данных

Вывод данных в C++ осуществляется с помощью команды **cout** (**C**onsole **Out**put).

Пространство имен

std — это пространство имён, определённое для всей стандартной библиотеки C++, а «::» — это оператор разрешения области видимости, который указывает, из какого пространства имён должен браться следующий за ним идентификатор.

Пространство имен

Пространство имён — группа идентификаторов, внутри которой все идентификаторы уникальны (не повторяются).

Пространство имен

С помощью идентификаторов, состоящих из латинских букв и цифр можно давать имена различным сущностям программы: переменным, функциям, методам, классам. С помощью разных пространств имён можно использовать одни и те же имена в одной и той же программе. Кроме того, пространство имён позволяет решить следующую программу: в собственных программах для создания сущностей мы можем использовать те же имена, что задействованы в сторонних библиотеках, в том числе, в стандартной библиотеке C++.

Пространство имен

Если отсутствует необходимость в использовании разных пространств имён в рамках одной программы, то можно однажды задать пространство и далее обращаться ко всем именам без его указания.

using namespace std;

Пространство имен

На всей стандартной библиотеке единое пространство имён std помогает объединить описанные в библиотеке ресурсы в единое целое. Стандартная библиотека устроена так, что в разных её файлах нет повторяющихся идентификаторов на одном уровне. Стандартная библиотека распределена по разным заголовочным файлам, каждый из которых мы можем подключить к программе, но пространство имён везде общее — std.

Библиотека iostream

```
#include <iostream>
using namespace std;
int main()
{
   setlocale(LC_ALL,””);
   int i;
   cout << "Введите целое число\n";
   cin >> i;
   cout << "Вы ввели число" << i << ", спасибо!";
return 0;
}
```

Библиотека iostream

Пример программы, выводящей на экран квадрат числа, введённого пользователем с клавиатуры:

```
#include <iostream>
using namespace std;
int main()
  {     setlocale(LC_ALL,””);
        cout << “Введите число: ";
        int s;
        cin >> s;
        cout << «Квадрат числа: ";
        cout << s*s << endl;
        return 0;
   }
```

Библиотека iostream

return 0

Ноль означает отсутствие ошибки.

Для функции main, даже объявляя её как возвращающую целое число, и только для неё, можно ничего не возвращать.

Библиотека iostream

Пример программы, выводящей на целую часть вещественного числа, введённого пользователем с клавиатуры:

```
#include <iostream>
using namespace std;
int main() {
        double num;
        cout << "Vvedite chislo: ";
        cin >> num;
        int ch = (int) num;
        cout << "Tselaya chast chisla: " << ch << endl;
        return 0;
}
```

Преобразование типов данных

В C++ различают два вида преобразования типов данных: явное и неявное.

- **Неявное преобразование** происходит автоматически. Это выполняется во время сравнения, присваивания или вычисления выражения различных типов.

Наивысший приоритет получает тот тип, при котором информация теряется менее всего. Не стоит злоупотреблять неявным преобразованием типов, так как могут возникнуть разного рода непредвиденные ситуации.

Преобразование типов данных

Явное приведение осуществляется с помощью указания целевого типа данных (того, к которому нужно привести) в круглых скобках перед выражением:

```
double s = 2.71;
int t = (int) s;
cout << t << endl; // 2
cout << (int) 3.14 << endl; // 3
cout << (int) (2.5 + t) << endl; // 4
```

Приведение к целым числам от вещественных осуществляется путём отбрасывания целой части (не округлением).

Преобразование типов данных

В C++ к тому же возможно приведение между логическим и числовыми типами.

Любое ненулевое число приводится к true, число 0 или 0.0 — к false. И, наоборот, false преобразуется в 0, а true — в 1.

```
bool b = true;
int t = (int) b;
cout << t << endl; // 1
cout << (int) false << endl; // 0
bool n = (bool) 42; // true
bool m =  (bool) (5 * 2 - 10); // false
bool k = (bool) 0.43; // true
double num1 = 2 + (double) (k || m); // 3.0
```

Библиотека math.h

- Чтобы воспользоваться сложными математическими действиями, нам нужно подключить в программу библиотеку, в которой и содержаться эти функции, а именно:

#include<math.h>

Библиотека math.h

Рассмотрим, какие функции содержатся в этой библиотеки.

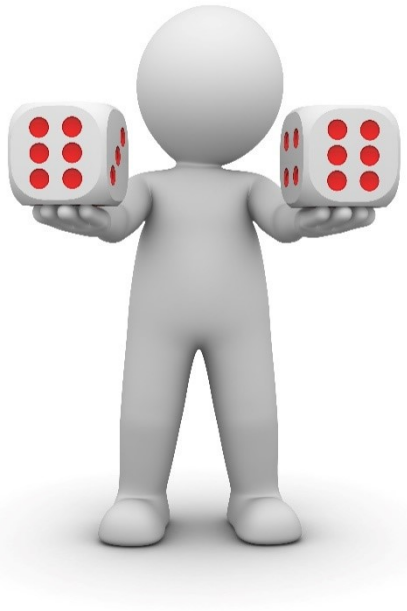
- abs – это модуль, возвращает положительное число
- acos (xxx)- арккосинус
- asin (sss) — арксинус
- atan (poiy) — арктангенс
- cos (sgrgrg) — косинус
- Random- вывод случайных чисел
- exp — экспонента
- log (56) — натуральный логарифм
- log10 (45,755) — это логарифм по основанию десять.
- pow(xx,yyy)- возведение в степень
- sin — синус
- tan — тангенс

Библиотека cmath

Имя функции	Описание
abs	Возвращает абсолютную величину (модуль) целого числа
acos	арккосинус
asin	арксинус
atan	арктангенс
ceil	округление до ближайшего большего целого числа
cos	косинус
cosh	гиперболический косинус
exp	вычисление экспоненты
fabs	абсолютная величина (для чисел с плавающей точкой)
floor	округление до ближайшего меньшего целого числа
fmod	вычисление остатка от деления нацело для чисел с плавающей точкой
log	натуральный логарифм
Log10	логарифм по основанию 10
pow(x,y)	результат возведения x в степень y, x^y
sin	синус
sinh	гиперболический синус
sqrt	квадратный корень
tan	тангенс
tanh	гиперболический тангенс

Также в C++ доступны две константы: число «пи» и число «е» (основание экспоненциальной функции или число Неппера). Их можно получить с помощью констант

M_PI и M_E



Литература

Васильев А.Н. Самоучитель C++ с примерами и задачами. Глава 1.