Язык C++

Часть I

ἀγεωμέτρητος μηδείς είσιτω

Определение 1. *Формальный язык — множество слов (цепочек символов), составленных из конечного множества символов* — алфавита.

Формальный язык может быть определён перечислением слов, входящих в данный язык. Обычно для этого:

- задается алфавит, из символов которого затем строятся все слова и выражения языка;
- описывается *синтаксис* языка, то есть *правила* построения осмысленных (допустимых для данного языка) выражений.

Определение 2. *Язык программирования — формальный язык, предназначенный для записи алгоритмов в виде компьютерных программ.*

Язык программирования, в отличие от естественных языков, предназначен для передачи команд и данных от человека к компьютеру, а не для общения людей между собой.

Язык программирования должен определять:

Лексику языка, т. е. его словарный состав.

Синтаксис языка — правила формирования слов и выражений программ из символов.

Семантику — смысловое значение слов выражений языка.

Языки программирования традиционно разделяют на языки *низкого уровня* (low-level languages) и языки *высокого уровня* (high-level languages).

Низкоуровневые языки, близкие к машинному коду, позволяют создать программы, которые (при умелом программировании!) работают быстрее и позволяют эффективнее использовать ресурсы компьютера. Язык низкого уровня, однако, привязан к устройству процессора, программы на нем не являются переносимыми на иную компьютерную архитектуру, так как тип процессора жёстко определяет набор доступных команд машинного языка. Языки низкого уровня, как правило, используют для написания небольших системных программ, драйверов устройств, кода специализированных микропроцессоров и т. п, когда важнейшими требованиями являются компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам. Пример: Ассемблер (assembler).

В языках высокого уровня особенности конкретных компьютерных архитектур не имеют значения, поэтому созданные приложения легко переносятся с компьютера на компьютер. Обычно достаточно просто перекомпилировать программу под определенную компьютерную архитектурную и операционную систему. Языки высокого уровня проще в использовании для программиста, но их эффективность ограничена.

Код простейших программ на ассемблере и C++.

```
; Hello World for Intel Assembler (MSDOS)
mov ax,cs
mov ds,ax
mov ah,9
mov dx, offset Hello
int 21h
xor ax,ax
int 21h
Hello:
  db "Hello World!",13,10,"$"
```

```
// Hello World C++
#include <iostream>
int main()
{
  std :: cout << "Hello World!" << std:: endl;
}
```

Определение 3. *Трансляция программы — преобразование программы, представленной на одном из языков программирования, в программу на другом языке.*

В частности, программы, написанные на языках высокого уровня, необходимо преобразовать в машинные (бинарные) коды, для того, чтобы они могли исполняться компьютером. Программа, выполняющая трансляцию, называется *транслятором* (translator — переводчик, от translate — переводить, переделывать).

Различают два основных способа трансляции — *компиляция* (compile — собирать вместе, составлять) программы или ее *интерпретация* (interpret). При компиляции весь *исходный программный код* (тот, который пишет программист) сразу переводится в машинный. Создается так называемый отдельный *исполняемый файл*, который выполняется операционной системой. При интерпретации выполнение кода происходит последовательно (строка за строкой). Операционная система взаимодействует со специальной программой — *интерпретатором*, а не с исходным кодом. Когда используется интерпретатор, он должен присутствовать все время выполнения программы. Выполнение скомпилированной программы происходит быстрее, т. к. она представляет собой готовый машинный код.

Для C/C++ процесс трансляции файлов исходного кода в исполняемый файл проходит в два основных этапа:

- При компиляции файлов исходного кода (.c, .cc, или .cpp) и создаются *объектные файлы* программы. Компилятор транслирует высокоуровневый код в машинный язык (расширение таких файлов зависит от компилятора, например .obj или .o). Каждый из этих файлов содержит машинные инструкции, которые эквивалентны исходному коду.
- При *компоновке* («линковке», от link — связь, соединение) из нескольких объектных файлов и, как правило, т. н. библиотечных файлов (подключаемых директивой #include <*.h>), создается единый исполняемый файл (.exe).

IDE (Интегрированная среда разработки) включает текстовый редактор для написания и редактирования исходных текстов программ, компилятор, компоновщик, отладчик и другие элементы. Разработка, компиляция и запуск программы осуществляется непосредственно в IDE. Интегрированные среды разработки существенно упрощают процесс составления программ, так как написание кода компиляция и запуск программ выполняются в одной программе, а также позволяет быстро найти и исправить ошибки.

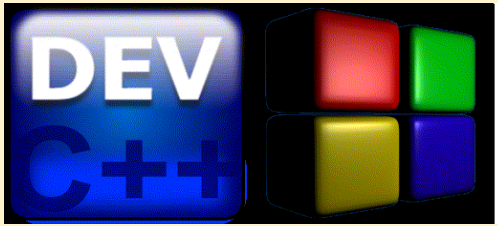
IDE для C++.

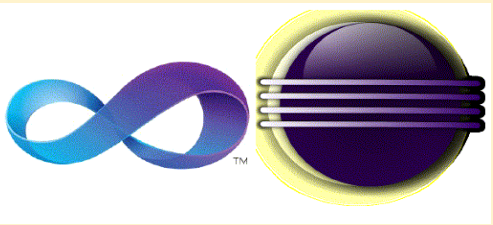
Dev-C++ — бесплатная IDE под Windows, функциональна и компактна, хорошо подходит для начинающих, `http://www.bloodshed.net/devcpp.html`.

Code::Blocks — бесплатная кросс-платформенная (Linux/Windows/Mac OS X) среда разработки, `http://www.codeblocks.org/`.

Microsoft Visual Studio Express — бесплатная версия популярной IDE Visual Studio, среда начального уровня для лиц не занимающихся профессионально программированием (учащихся, студентов, любителей и т. д.), `http://www.microsoft.com/express/`.

Eclipse C/C++ Development Tools — популярная бесплатная кросс-платформенная (FreeBSD/Linux/Mac OS X/Solaris/QNX/Windows) среда разработки, `http://www.eclipse.org/cdt/`.





Язык C++ разработан в начале 1980-х годов Бьерном Страуструпом (Bjarne Stroustrup) на основе языка C. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков, в отличие от своего предшественника, языка C, поддерживает *объектно-ориентированное* программирование. В настоящее время C++ является одним из самых популярных языков программирования, широко используется для разработки операционных систем, разнообразных прикладных программ, драйверов устройств, высокопроизводительных серверов, а также компьютерных игр.

Рис. 1. Бьерн Страуструп

В простейшем случае структуру С-программы можно представить на примере

```
#include   ...    // подключение
#include   ...    // необходимых библиотек
    ..............
/* глобальные переменные: */
int        k = 10;
    ..............
/* функции: */
char  f1()  {...}
    ..............
int  main()
{
    ..............
  return  0;
}
```

Из всех указанных разделов необходимым является только главная функция main(), остальные — необязательны.

Программа на C++ может состоять из любого количества функций, одна из которых должна называться main(). Этой функции передается управление сразу после запуска программы.

Текст между символами /* */ игнорируется компилятором и называется *комментарием*, он предназначен для программиста. Другой вид комментария — однострочный, начинается с //

```
/* комментарий на нескольких
  строках  */
//  комментарий до конца строки
```

Каждый оператор[1] программы должен заканчиваться ;. Программа пишется в свободной форме, поэтому одной строке можно писать сколько угодно операторов, но для лучшей читаемости листингов рекомендуется располагать их на отдельных строках. Также желательно придерживаться единого стиля форматирования текста.

Фигурные скобки { } объединяют несколько операторов в группу, именуемую *блоком*.

[1]Оператор (statement) — инструкция, команда, например: k = 10 — «k присвоить 10». Часто словом «оператор» в программировании ошибочно называют *операцию* (operator) — специальный способ записи некоторых действий, напр. для сложения: «+».

Имена переменных, констант, функций, классов и т. п. можно записывать, используя любые буквы латинского алфавита, символ _ и арабские цифры, причем начинаться имя должно с буквы.

Замечание 1. *Компилятор различает прописные и строчные буквы.* Поэтому имена name, Name, naME будут считаться различными.

Ввод/вывод данных на консоль.

Файл iostream.* предназначен для поддержки операций ввода/вывода, доступ к его функциям осуществляется при подключении *заголовочного* (header) файла iostream.h с помощью директивы #include <iostream.h>. В программе создаются «каналы связи» cin для ввода с клавиатуры и cout для вывода на экран (*консоль*), а также операции чтения/записи в *поток* (stream).

С помощью объекта cin и операции >> можно присвоить значение любой переменной, например, оператор cin >> k присвоит переменной k значение, введенное с клавиатуры.

Объект cout и операция << позволяет вывести на экран значение любой переменной или текст. Текст необходимо заключать в двойные кавычки (русский текст может выводиться неправильно). Команда cout << n приводит к выводу на экран значения переменной n.

Если не указано пространство имен **using namespace** std;, то перед всеми объектами следует указывать префикс std::.

```
int _tmain( int argc, _TCHAR* argv[] )
{
  int k = 0;                    // объявление переменной k
  std :: cout << " k = ";       // приглашение к вводу
  std :: cin >> k;              // ввод значения для k
  std :: cout << "10k = " << 10*k << std::endl;
  getchar ();                   // предотвращает закрытие консоли
  getchar ();                   // до нажатия <Enter>
  return 0;
}
```

Можно использовать в cout специальные символы для табуляции или перехода на новую строку и даже для воспроизведения звука, отображать числа в десятичном, восьмеричном или шестнадцатеричном формате и т. п.

Поскольку вывод в консольное окно выполняется в кодировке ASCII, которая использовалась в старых операционных системах (MS-DOS), а текст программы набран в другой кодировке, то вывод кириллицы в консоль Windows вызывает проблемы.

Определение 4. *Переменная — это именованная область памяти, в которой хранятся данные заданного типа.*

Перед первым использованием в программе любую переменную необходимо *объявить*. Во время выполнения программы значение переменной может быть изменено.

Объявление переменных. Формат записи:

```
тип_переменной имя_переменной;
```

При определении переменной ей может быть присвоено начальное значение:

```
тип_переменной имя_переменной = значение_переменной;
```

Замечание 2. *Настоятельно рекомендуется всегда совмещать объявление переменной и присвоение ей начального значения.*

В одном операторе можно описать несколько переменных одного типа, разделяя их запятыми (не рекомендуется).

Если переменная определена внутри блока ({ }), она называется *локальной*, область ее действия (видимости) — от точки описания до конца блока, включая все вложенные блоки. Если переменная определена вне любого блока, она называется *глобальной* и областью ее действия считается файл, в котором она определена, от точки описания до его конца. *Пример...*

От типа данных зависит:

- Объем памяти, выделяемый под данные.
- Представление данных в памяти компьютера.
- Множество допустимых операций.
- Множество допустимых значений.

Простые типы данных в C++ — целые, вещественные, символьный, логический тип и тип void. На основе простых типов строятся более сложные типы: массивы, структуры, классы и др.

Целый тип предназначен для представления в памяти компьютера целых чисел. Основной целый тип — **int**. Имеет разновидности: **short** (короткое целое) и **long** (длинное целое), а также к целым типам относится символьный тип **char**. Для описания беззнаковых целых переменных добавляется спецификатор **unsigned**.

Примеры:

```
int          i , day , month , year ;
long int     miriada ;
char         bukva;
```

При объявлении переменных можно (и настоятельно рекомендуется!) провести их *инициализацию* (присвоить им начальное значение), например

```
unsigned int    month = 12;
char            bukva = 'W';
```

В C++ объявление и инициализация разрешены практически в любом месте программы. Переменная любого типа может быть объявлена как далее немодифицируемая (константа)

```
const int     m = 1234567;
```

после чего ей нельзя присваивать другое значение.

Тип **char** используется для представления символов в соответствии с кодировкой ASCII (American Standard Code for Information Interchange — Американский стандартный код обмена информации) — восьмибитный код, с помощью которого можно представить 256 различных символов.

Логический тип (**bool**) Величины логического типа могут принимать только значения true (истина) и false (ложь). Внутренняя форма представления значения false — 0. Любое ненулевое значение трактуется как true.

Числа с плавающей точкой для представления вещественных чисел: **float** и **double**. Внутреннее представление вещественного числа состоит из *мантиссы* и *порядка*. Длина мантиссы определяет точность числа, а величина порядка — его диапазон.

тип	диапазон значений	размер (байт)
bool	true, false	1
char	0 . . . 255	1
signed short int	−32768 . . . 32767	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	$3{,}4 \cdot 10^{-38} \ldots 3{,}4 \cdot 10^{38}$	4
double	$1{,}7 \cdot 10^{-308} \ldots 1{,}7 \cdot 10^{308}$	8

Тип **void** (пустой) используется, в частности, для определения функций, которые не возвращают значения.

Арифметические операторы

```
+, −, *, /
%                          // деление по модулю
```

Оператор присваивания = (читать: присвоить)

```
имя_переменной = выражение;
```

Сначала вычисляется значение выражения в правой части, затем оно присваивается переменной («старое» значение переменной при этом теряется). Примеры:

```
int  var1 = 5 + 3;
int  var2 = 20 % var1;  // var2 = ?
var1 = var1 + var2;     // var1 = ?
```

Составные операторы присваивания +=, *=, %= и т.д.

```
int n = 11;
n += 100;          // сокращение  n = n + 100;
n %= 3;            // сокращение  n = n % 3;
```

Инкремент и декремент.

```
int n = 11;
++n;          // n <-- 12
--n;          // n <-- 11
n++;          // n <-- 12
```

Замечание 3. *Если операнды арифметических операций имеют один тип, то и результат операции будет иметь тот же тип. Поэтому оператор /, примененный к к целым будет отбрасывать остаток, например, 1 / 2 == 0. Если нужно именно вещественное деление 1.0 / 2 == 0.5.*

Замечание 4. *При расчетах следует помнить о диапазонах типов. Пример целочисленного переполнения:*

```
int _tmain( int argc, _TCHAR* argv[] )
{
 unsigned short int n = 65536; // unsigned short int : 0...65535
 cout << " n = " << n;
 getchar ();
 return 0;
}
```

Операторы отношения. Результатом этих операций являются логические значения, true (истина) или false (ложь).

```
==              // сравнение на равенство, читать: равно
!=              // не равно
<, >, <=, >=    // неравенства
```

Логические операторы.

```
&&              // логическое И
||              // логическое ИЛИ
 !              // отрицание, НЕ
```

Замечание 5. *Операторы, имеющие более высокий* приоритет *(приоритет вычисления) выполняются раньше, чем операторы с более низким приоритетом. Например, умножение и деление выполняется раньше сложения и умножения. Для изменения приоритетного порядка вычислений, так же как в математике, можно использовать скобки (). Таблицу приоритетов всех операторов можно найти в справочниках по языку, например — [4].*

Задачи

1. В банк делается вклад x руб. под 7% годовых. Каков будет банковский счет через 3 года?

Ввод: число x типа **double**, вывод: число того же типа.

2. Определить сумму цифр трехзначного натурального числа.

Ввод: число типа **unsigned short int**, вывод: сумма его цифр.

3. Что будет напечатано? Почему?

```
char a = '1';
char b = '2';
char x = a + b;
cout << " x = " << x <<endl;  // ??
```

4. Считается, что человек с весом v кг может выпить без вреда для здоровья v г спирта. Сколько человек с весом v кг может выпить водки? (Без вреда для здоровья). Считать вес спирта равным весу воды того же объема.

Ввод: число v типа **unsigned short int**, вывод: число типа **double** — водка в граммах.

Список литературы

1. Герберт Шилдт. C++: базовый курс. М.: Вильямс, 2008.
2. Стивен Прата. Язык программирования C++. Лекции и упражнения, 5-е изд. М.: Вильямс, 2007.
3. Бьерн Страуструп. Язык программирования C++. М.: Бином, 2011
4. Герберт Шилдт. Полный справочник по C++, 4-е изд. М.: Вильямс, 2006.