Введение в язык программирования C++

История создания языка C и C++

Язык C был создан в начале 70х годов XX века **Кеном Томпсоном** и **Дэннисом Ритчи** из Bell Labs.

Язык C использовался для создания операционной системы UNIX.

C – структурный язык программирования, но в нем можно создавать собственные абстрактные типы данных и давать другие названия существующим типам.

Стандартная международная версия языка появилась в 1990 году.

Язык C++ был разработан **Бьерном Страуструпом** в 1979 году. К языку C были добавлены возможности работы с классами и объектами (по аналогии с языком Симула).

Структура программы на языке C++

Простейшая программа на языке C++ имеет следующую структуру:

```
директивы препроцессора
void main()
{
операторы функции
}
```

Все директивы препроцессора начинаются со знака #. В конце каждого оператора ставится точка с запятой.
Примеры директив препроцессора:

```
#include <stdio.h>
#include "func.c"
```

Для организации ввода-вывода в языке C++ используется библиотека iostream, но можно использовать также функции из стандартных библиотек языка C (библиотека stdio.h)

Идентификаторы, ключевые слова, комментарии

Идентификаторы используются , чтобы обращаться к программным объектам. В идентификаторе могут использоваться буквы, цифры и знак подчеркивания. Первым символом не может быть цифра. Прописные и строчные буквы различаются.
sysop, SySoP и SYSOP — три разных имени

Ключевые слова – это зарезервированные идентификаторы, которые имеют специальное значение для компилятора. Их нельзя использовать в качестве имен программных объектов. Например, new, using, false и т.д.

Комментарии предназначены для записи пояснений к программе и формирования документации.

Однострочный комментарий начинается с //
Многострочные комментарий заключается между символами-скобками /* и */

Скалярные типы данных. Константы

Тип данных определяет:
- внутреннее представление данных в памяти компьютера;
- множество значений, которые могут принимать величины этого типа;
- операции и функции, которые можно применять к величинам этого типа.

В языке C++ шесть основных типов, на основе которых программист может вводить описание составных типов. Их называют **арифметическими типами.**

int – целый
char – символьный
wchar_t – расширенный символьный
bool - логический
float – вещественный
double – вещественный с двойной точностью

Спецификаторы типов:
short – короткий
long – длинный
signed – знаковый
unsigned – беззнаковый

Спецификатор указывается перед названием типа.

Типы short int, long int, signed int и unsigned int можно сокращать до short, long, signed и unsigned соответственно.

Вещественные типы:
float – 4 байта (порядок 8 разрядов)
double – 8 байт (порядок 11 разрядов)
long double – 10 байт (порядок 15 разрядов)

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

Константа – это неизменяемая величина.

Типы констант:

Константа	Формат	Примеры
Целая	Десятичный: последовательность десятичных цифр, начинающаяся не с нуля, если это не число нуль	8, 0, 199226
	Восьмеричный: нуль, за которым следуют восьмеричные цифры (0,1,2,3,4,5,6,7)	01, 020, 07155
	Шестнадцатеричный: 0x или 0X, за которым следуют шестнадцатеричные цифры (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)	0xA, 0x1B8, 0X00FF
Вещественная	Десятичный: [цифры].[цифры][2]	5.7, .001, 35.
	Экспоненциальный: [цифры][.][цифры]{E|e}[+|-][цифры][3]	0.2E6, .11e–3, 5E10
Символьная	Один или два символа, заключенных в апострофы	'A', 'ю', '•', 'db', '\0', '\n', '\012', '\x07\x07'
Строковая	Последовательность символов, заключенная в кавычки	"Здесь был Vasia", "\t3начение r=\0xF5\n"

Вещественные константы имеют по умолчанию тип double/

Символьные константы, состоящие из одного символа, занимают в памяти 1 байт и имеют стандартный тип char.

Символ \ (обратная косая черта) используется для представления:
-кодов, не имеющих графического изображения (\n - перевод курсора в начало следующей строки);
-символов апострофа, обратной косой черты, вопроса и кавычек;
-любого символа, заданного с помощью его 16-ричного или 8-ричного кода (\073, \0xA5)

Последовательности символов, начинающиеся в обратной косой черты, называют **управляющими или escape-последовательностями**.

Строковые константы, отделенные в программе только пробелами, при компиляции объединяются в одну. Длинную строковую константу можно разместить в нескольких строках:
“Пермский \
университет”

В конец каждого строкового литерала компилятором добавляется нулевой символ, который представляется последовательность \0.

Если тип константы, используемый по умолчанию, не устраивает программиста, он может явно указать требуемый тип с помощью суффиксов L, l, U, u. Например:
32L, 0x22UL

При описании типа переменной можно сразу же присвоить начальное значение:
int a,b=4,c;

Операции ввода-вывода

В C++ нет встроенных в язык средств ввода-вывода. Для этих целей используется библиотека ввода-вывода **iostream.**

```
#include <iostream>
```

Библиотека iostream определяет несколько стандартных потоков:

- **cin** стандартный входной поток
- **cout** стандартный выходной поток
- **cerr** и **clog** – потоки для вывода сообщений об ошибках (cerr выводит сообщения немедленно, clog – после заполнения буфера)

Стандартные потоки по умолчанию привязаны к консоли, но можно их перенаправить на другие устройства или файлы.

Стандартные потоки принадлежат пространству имен **std.**

Для выполнения операций ввода-вывода переопределены две операции:
>> получить из входного потока
<< поместить в выходной поток.

Ввод значения переменной:
cin >> идентификатор;
cin >> переменная1 >> переменная2 >>...>> переменная n;

Например:

```
int n;
char j;
cin >> n >> j;
```

Вывод информации:
cout << значение;
cout <<значение1 <<значение2 << ... << значение n;

Например:

```
int n;
char j;
cin >> n >> j;
cout << "Значение n равно" << n << "j=" << j;
```

Пространство имен - это группа имен, в которой имена не совпадают. Имена в различных пространствах имен не конфликтуют между собой, даже если они совпадают. Пространства имен вводятся для снижения вероятности конфликта имен и полезны в случае использования имен из нескольких различных библиотек.

Чтобы явно указать, а каком пространству имен принадлежит та или иная функция, необходимо указывать название пространства имен перед ее вызовом, то есть

```
std::cout << "Введите целое число\n";
std::cin >> i;
std::cout << "Вы ввели число " << i << ", спасибо!";
```

Чтобы не указывать одно и то же имя пространства имен несколько раз, можно однократно использовать инструкцию

```
using namespace имя_пространства имен;
```

Для использования русского шрифта при выводе сообщений необходимо использовать дополнительные функции:
setlocale (LC_ALL, "Russian") - для Visual C++ 2010
SetConsoleCP(1251) и SetConsoleOutputCP(1251) для Visual C++ 2005

Для их использования необходимо подключить библиотеку windows.h.

Для изменения формата вывода текста используются специальные функции, называемые манипуляторами. Для их использования нужно подключить библиотеку **iomanip**.

Основные манипуляторы:

endl	Вывод символа новой строки (переход в новую строку)
setprecision(int p)	Устанавливает число цифр после запятой при выводе
setw(int w)	Устанавливает ширину поля равной w
fixed	Вывод числа в форме с фиксированной точкой
scientific	Вывод числа в экспоненциальной форме

Пример:

```
#include <iostream>
#include <iomanip>
using namespace std;
void main()
{
cout << setprecision (2) << 1000.243 << endl;
cout << setw (20) << "Hello there.";
}
```

Программа выводит следующие данные:

```
1000.24
    Hello there.
```

Подробнее про манипуляторы и флаги ввода-вывода можно прочитать здесь - http://kvodo.ru/urok-10-formatirovannyiy-vvod-vyivod-v-c.html

Основные операции C++

Операнд - это переменное или постоянное значение, которым оперирует процессор в процессе реализации программных вычислений. Это могут быть константы, переменные, функции, выражения и другие объекты языка программирования, над которым производится операция.

Операция - это символ, представляющий собой некоторую операцию, производимую с данными.

Унарные операции

1) Операции увеличения и уменьшения значения на 1 (++ и --). Эти операции, называемые также **инкрементом** и **декрементом**, имеют две формы записи - префиксную, когда операция записывается перед операндом, и постфиксную

```
int x = 3; y = 3, z;
z=++x;
cout<<z; //4
z=y++;
cout<<z; //3
cout<<x;  //4
cout<<y; //4
}
```

2) **Операция определения размера sizeof** предназначена для вычисления размера объекта или типа в байтах, и имеет две формы:

sizeof (выражение)

sizeof (тип)

Например:

```
int x,y;
x=2; y=3;
cout<<sizeof(int);
cout<<sizeof(x+y);
```

3) **Операции отрицания (-, !** и **~**) – арифметическое отрицание (операция смены знака), логическое отрицание, поразрядное отрицание.

Бинарные операции

1) Деление (/) и остаток от деления (%).

```
#include <iostream>
void main()
{
int x = 11, y = 4;
float z = 4;
cout<<x/y<<x/z<<endl;
cout<<x%y;
}
Результат работы программы:
Результаты деления: 2 2.750000
Остаток: 3
```

Чтобы привести результат вычислений к нужному типу при целочисленном делении, кроме использования форматного вывода можно использовать операции приведения **float(x)** или **double(x)**.

```
#include <iostream>
using namespace std;
void main()
{
int x = 11, y = 4;
float z = 4, r1, r2;
r1=float(x)/y;
r2=float(x)/z;
cout << "Результаты деления: " << r1 ,<< r2 << endl;
}
```

2) **Операции сдвига (<<и >>)** применяются к целочисленным операндам. Они сдвигают двоичное представление первого операнда влево или вправо на количество двоичных разрядов, заданное вторым операндом

3) **Операции отношения (<, <=, >, >=, ==, !=)** сравнивают первый операнд со вторым.

4) **Поразрядные операции (&, |, ^)** применяются только к целочисленным операндам и работают с их двоичными представлениями. При выполнении операций операнды сопоставляются побитово (первый бит первого операнда с первым битом второго, второй бит первого операнда со вторым битом второго, и т д.).

5) **Логические операции (&& и ||).**

6) **Операции присваивания (=, +=, -=, *= и т. д.).**
Примеры:

```
b-=2;   //b=b-2
c*=b;   //c=c*b;
```

7) **Формат операции простого присваивания (=):**
операнд_1 = операнд_2

Тернарная операция

операнд_1 ? операнд_2 : операнд_3

Если результат вычисления операнда 1 равен true, то результатом условной операции будет значение второго операнда, иначе - третьего операнда

```
#include <iostream>
void main()
{
int a = 11, b = 4, max;
max = (b > a)? b : a;
cout<<max;
}
Результат работы программы:
Наибольшее число: 11
```