**Начальные сведения о языке**

### История и назначение языка Си++

Разработчиком языка *Си*++ является *Бьерн Страуструп*. В своей работе он опирался на *опыт* создателей языков Симула, Модула 2, абстрактных типов данных. Основные работы велись в исследовательском центре компании *Bell* Labs.

Непосредственный предшественник *Си*++ – язык *Си* с классами – появился в 1979 году, а в 1997 году был принят *международный стандарт* *Си*++, который фактически подвел итоги его 20-летнего развития. Принятие стандарта обеспечило единообразие всех реализаций языка *Си*++. Не менее важным результатом стандартизации стало то, что в процессе выработки и утверждения стандарта язык был уточнен и дополнен рядом существенных возможностей.

На сегодня стандарт утвержден Международной организацией *по* стандартизации *ISO*. Его номер *ISO/IEC 14882*. *ISO* бесплатно стандарты не распространяет. Его можно получить на узле американского национального комитета *по* стандартам в информационных технологиях: [www.ncits.org](http://www.ncits.org/)

В России следует обращаться в ВНИИ Сертификации:

[http://www.vniis.ru](http://www.vniis.ru/)

Проекты стандарта имеются в свободном доступе:

<http://www.research.att.com/~bs/bs_faq.html>

Язык *Си*++ является универсальным языком программирования, в *дополнение* к которому разработан набор разнообразных библиотек. Поэтому, строго говоря, он позволяет решить практически любую задачу программирования. Тем не менее, в силу разных причин (не всегда технических) для каких-то типов задач он употребляется чаще, а для каких-то – реже.

*Си*++ как преемник языка *Си* широко используется в системном программировании. На нем можно писать высокоэффективные программы, в том числе операционные системы, драйверы и т.п. Язык *Си*++ – один из основных языков разработки трансляторов.

Поскольку системное *программное обеспечение* часто бывает написано на языке *Си* или *Си*++, то и программные интерфейсы к подсистемам ОС тоже часто пишут на *Си*++.

Распределенные системы, функционирующие на разных компьютерах, также разрабатываются на языке *Си*++. Этому способствует то, что у широко распространенных компонентных моделей *CORBA* и *COM* есть удобные интерфейсы на языке *Си*++.

Обработка сложных структур данных – текста, бизнес-информации, *Internet*-страниц и т.п. – одна из наиболее распространенных возможностей *применения* языка. В прикладном программировании, наверное, проще назвать те области, где язык *Си*++ применяется мало.

Разработка графического пользовательского интерфейса на языке *Си*++ выполняется, в основном, тогда, когда необходимо разрабатывать сложные, нестандартные интерфейсы. Простые программы чаще пишутся на языках *Visual* *Basic*, *Java* и т.п.

*Программирование* для *Internet* в основном производится на языках *Java*, VBScript, Perl.

В целом надо сказать, что язык *Си*++ в настоящее время является одним из наиболее распространенных языков программирования в мире.

### Простейшая программа на языке Си++

Самая короткая *программа* на языке *Си*++ выглядит так:

// Простейшая программа

int main() { return 1; }

Первая строчка в программе – комментарий, который служит лишь для пояснения. Признаком комментария являются два знака деления подряд ( // ).

main – это имя главной функции программы. С функции main всегда начинается выполнение. У функции есть имя ( main ), после имени в круглых скобках перечисляются аргументы или параметры функции (в данном случае у функции main аргументов нет). У функции может быть результат или возвращаемое *значение*. Если *функция* не возвращает никакого значения, то это обозначается ключевым словом void. В фигурных скобках записывается *тело функции* – действия, которые она выполняет. Оператор return 1 означает, что *функция* возвращает результат – *целое число* 1.

Если мы говорим об объектно-ориентированной программе, то она должна создать *объект* какого-либо класса и послать ему сообщение. Чтобы не усложнять программу, мы воспользуемся одним из готовых, предопределенных классов – классом iostream (*поток ввода-вывода*, *базовый класс* для iostream). Этот *класс* определен в *файле* заголовков " iostream.h ". Поэтому первое, что надо сделать – включить *файл* заголовков в нашу программу:

#include <iostream.h>

int main() { return 1; }

Кроме класса, *файл* заголовков определяет глобальный *объект* этого класса cout. *Объект* называется глобальным, поскольку *доступ* к нему возможен из любой части программы. Этот *объект* выполняет *вывод* на *консоль*. В функции main мы можем к нему обратиться и послать ему сообщение:

#include <iostream.h>

int main()

{

 std::cout << "Hello, world!" << std::endl;

 return 1;

}

Операция сдвига << для класса iostream определена как "вывести". Таким образом, *программа* посылает объекту cout сообщения "вывести строку Hello, world!" и "вывести *перевод строки*" ( endl обозначает новую строку). В ответ на эти сообщения *объект* cout выведет строку " Hello, world!" на *консоль* и переведет *курсор* на следующую строку.

### Компиляция и выполнение программы

*Программа* на языке *Си*++ – это текст. С помощью произвольного текстового редактора программист записывает инструкцию, в соответствии с которой *компьютер* будет работать, выполняя данную программу.

Для того чтобы *компьютер* мог выполнить программу, написанную на языке *Си*++, ее нужно перевести на язык машинных инструкций. Эту задачу решает *компилятор*. *Компилятор* читает *файл* с текстом программы, анализирует ее, проверяет на предмет возможных ошибок и, если таковых не обнаружено, создает исполняемый *файл*, т.е. *файл* с машинными инструкциями, который можно выполнять.

Откомпилировав программу один раз, ее можно выполнять многократно, с различными исходными данными.

Не имея возможности описать все варианты, остановимся только на двух наиболее часто встречающихся.

### Компилирование и выполнение программ в среде Windows

Если Вы используете персональный *компьютер* с операционной системой Microsoft  *Windows* 98 , *Windows* NT  или *Windows* 2000 , то *компилятор* у Вас, скорее всего, *Visual C++* . Этот *компилятор* представляет собой интегрированную среду программирования, т.е. объединяет *текстовый редактор*, *компилятор*, отладчик и еще ряд дополнительных программ. Мы предполагаем, что читатель работает с версией 5.0 или старше. Версии младше 4.2 изучать не имеет смысла, поскольку реализация слишком сильно отличается от стандарта языка.

В среде *Visual C++* прежде всего необходимо создать новый проект. Для этого нужно выбрать в *меню* File *атрибут* New. Появится новое диалоговое окно. В закладке Projects в списке различных типов выполняемых *файлов* выберите Win32 Console Application. Убедитесь, что отмечена кнопка Create new workspace. Затем следует набрать имя проекта (например, test ) в *поле* Project name и имя каталога, в котором будут храниться все *файлы*, относящиеся к данному проекту, в *поле* Location. После этого нажмите кнопку " OK ".

Теперь необходимо создать *файл*. Опять в *меню* File выберите *атрибут* New. В появившемся диалоге в закладке File отметьте text file. *По* умолчанию новый *файл* будет добавлен к текущему проекту test, в чем можно убедиться, взглянув на *поле* Add to project. В *поле* Filename нужно ввести имя *файла*. Пусть это будет main.cpp. Расширение .cpp – это стандарт для *файлов* с исходными текстами на языке *Си*++. *Поле* Location должно показывать на каталог C:\Work. Нажмите кнопку " OK ".

На экране появится пустой *файл*. Наберите текст программы.

*Компиляция* выполняется с помощью *меню* Build. Выберите *пункт* Build test.exe (этому пункту *меню* соответствует функциональная клавиша F7 ). В нижней части экрана появятся сообщения компиляции. Если Вы сделали опечатку, *двойной щелчок* мышью *по* строке с ошибкой переведет *курсор* в окне текстового редактора на соответствующую строку кода. После исправления всех ошибок и повторной компиляции система выдаст сообщение об успешной компиляции и компоновке (пока мы не будем уточнять, просто вы увидите сообщение Linking ).

Готовую программу можно выполнить с помощью *меню* Build, *пункт* Execute test.exe. То же самое можно сделать, нажав одновременно клавиши CTRL и F5. На экране монитора появится консольное окно, и в нем будет выведена строка " Hello, world!". Затем появится надпись "Press any key to *continue*". Эта надпись означает, что *программа* выполнена и лишь ожидает нажатия произвольной клавиши, чтобы закрыть консольное окно.

### Компилирование и выполнение программ в среде Unix

Если Вы работаете в операционной системе *Unix,* то, скорее всего, у Вас нет интегрированной среды разработки программ. Вы будете пользоваться любым доступным текстовым редактором для того, чтобы набирать тексты программ.

Редактор Emacs предпочтительнее, поскольку в нем есть специальный режим редактирования программ на языке *Си*++. Этот режим включается автоматически при редактировании *файла* с именем, оканчивающимся на " .cpp " или " .h ". Но при отсутствии Emacs сгодится любой *текстовый редактор*.

Первое, что надо сделать – это поместить текст программы в *файл*. В редакторе следует создать *файл* с именем main.cpp (расширение cpp используется для текстов программ на языке *Си*++). Наберите текст программы из предыдущего параграфа и сохраните *файл*.

Теперь программу надо откомпилировать. *Команда* вызова компилятора зависит от того, какой *компилятор* *Си*++ установлен на компьютере. Если используется *компилятор* *GNU* C++, *команда* компиляции выглядит так:

gcc main.cpp

Вместо gcc может использоваться g++, c++, cc. Уточнить это можно у системного администратора. Отметим, что у широко распространенного компилятора *GNU* C++ есть ряд отличий от стандарта *ISO*.

В случае каких-либо ошибок в программе *компилятор* выдаст на *терминал* сообщение с указанием номера строки, где обнаружена ошибка. Если в программе нет никаких опечаток, *компилятор* должен создать исполняемый *файл* с именем a.out. Выполнить его можно, просто набрав имя a.out в ответ на подсказку интерпретатора команд:

a.out

Результатом выполнения будет *вывод* на экран терминала строки:

Hello, world!

**Самостоятельная работа:**

Служебные слова Си++