Первое знакомство с программой C++

Структура единицы трансляции (текстовый файл исходного кода)

```
//   глобальная область видимости
//        директивы препроцессора C
//        объявление объектов программы:
//             переменных (по умолчанию статических)
//             функций
//             пространств имён
//             классов (объектно-ориентированное программирование)
//             шаблонов функций (обобщённое программирование)
//             шаблонов классов (обобщённое программирование)
//        определение объектов программы:
//             констант
//             функций
//   точка входа в программу (определение функции main())
        int main()
        // составная-инструкция
        {
             //  локальная область видимости
             return 0;    // инструкция-перехода
        }
```

Примечание.

Началом однострочного комментария в C++ является последовательность символов //.

Программирование в стиле С

```
/* Программирование в стиле С */

#include <stdio.h>
// подключаемый к единице трансляции ресурс
// в виде заголовочного файла

int main()
{
    printf("Hello, world!\n");    // инструкция-выражение
    return 0;
}
```

Примечание.

Признаком многострочного комментария являются два его ограничителя: началом комментария служит последовательность символов /*, а соответственно, его концом –*/.

Директива препроцессора С ***#include*** позволяет включить заголовочный файл ***<stdio.h>*** стандартной библиотеки С в исходный файл с исполняемым кодом. Угловые скобки в именовании заголовочных файлов указывают компилятору на стандартное размещение этих файлов в системных директориях. В заголовочном файле ***<stdio.h>*** представлены объявления функций ввода-вывода, в частности, используемой здесь библиотечной функции форматного вывода ***printf*****()**: ***int printf(const char*, …);***.

Аргументом вызова функции ***printf*****()** является строковый литерал ***"Hello, world!\n"***. Одним из символов строкового литерала является escape-последовательность ***\n*** – символ новой строки (от слов ***new line***), благодаря которому и осуществляется переход на новую строку.

Благодаря инструкции-перехода ***return*** реализуется механизм возврата в точку вызова результирующего значения ***0*** функции ***main*****()**.

Программирование в стиле C++

```
// Программирование в стиле C++ - первый шаг

#include <cstdio>

int main()
{
    printf("Hello, world!\n");
    return 0;
}
```

Примечание.

Средства стандартных библиотек C и C++ представлены набором заголовочных файлов. Для стандартных заголовочных файлов в библиотеке C обязательным атрибутом является расширение ***.h***, в то же время для указания стандартных заголовочных файлов библиотеки C++ этого расширения не требуется.

Стандартный заголовочный файл в библиотеке C++, имя которого начинается с буквы ***c*** (например, **<*cstdio*>**), эквивалентен заголовочному файлу в стандартной библиотеке C (например, **<*stdio.h*>**).

Средства стандартной библиотеки C++ определены в пространстве имен ***std***, в то время как средства стандартной библиотеки C определены в глобальном пространстве имен.

```
// Программирование в стиле C++ - второй шаг

#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
    return 0;
}
```

Примечание.

По умолчанию каждая программа на C++ может пользоваться стандартными консольными потоками: стандартным вводом (***cin***) и стандартным выводом (***cout***), а также стандартным выводом ошибок (***cerr*** и ***clog***). Эти потоки представляют собой последовательности символов (например, обычных ***char*** или расширенных ***wchar_t***).

Стандартные потоки ввода-вывода ***cin***, ***cout***, ***cerr*** и ***clog*** определены в пространстве имен ***std*** и представлены заголовочным файлом ***<iostream>***.

Для понимания основополагающих принципов организации стандартной библиотеки C++ наряду с представлением об императивной и модульной парадигмах программирования требуется также представление и о двух других парадигмах – объектно-ориентированной и обобщенной. Это необходимо для понимания таких сущностей, как класс, функция-член класса, операторная функция, функция-шаблон и класс-шаблон.

Одной из важных концепций, принятых при построении стандартной библиотеки потоков, является реализация операций ввода-вывода объектов встроенных типов в виде перегруженных стандартных операторов сдвига C++: оператора сдвига вправо **>>** (“прочесть из”) и оператора сдвига влево **<<** (“записать в”). Перегруженные операторы сдвига **>>** и **<<** представляют собой операторные функции классов ***istream*** и ***ostream*** и, соответственно, называются операторами ввода и вывода.

Операция вывода строкового литерала ***”Hello, world!\n”*** осуществляется с помощью инструкции-выражения. Выражение составлено из бинарного оператора вывода **<<** и двух его операндов – потока и строкового литерала. Чтобы указать на принадлежность имени потока ***cout*** пространству имен ***std***, применяется механизм квалификации имени ***cout*** с помощью унарного оператора разрешения области видимости **::**.

```
// Программирование в стиле C++ - третий шаг

#include <iostream>

int main()
{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Примечание.

На этапе первого знакомства с библиотекой потоков C++, как правило, не требуется явного привлечения механизма форматирования ввода-вывода, достаточно положиться на умолчание.

Управление форматированием ввода-вывода производится средствами класса ***basic_ios*** и его базового класса ***ios_base*** (набор флагов формата и операций для их установки и сброса). Для управления состоянием потока также можно воспользоваться и более изящными средствами стандартной библиотеки – манипуляторами – набором функций для манипулирования состоянием потока. Ключевая идея заключается в том, чтобы вставлять между объектами (читаемыми из потока или записываемыми в поток) операцию, которая изменяла бы состояние потока. Стандартные манипуляторы определены в пространстве имен ***std*** и представлены заголовочными файлами ***<iostream>*** и ***<iomanip>***.

Стандартный манипулятор ***endl*** (от слов ***end of line***) выводит в поток символ новой строки ***\n*** и очищает буфер потока.

Ранее уже отмечалось, что оператор ввода >> и оператор вывода << представляют собой операторные функции классов ***istream*** и ***ostream***. Так как возвращаемым значением этих функций является ссылка на ***istream*** или ***ostream***, то это как раз и позволяет применять к ней друг за другом все последующие операторы ввода или вывода в инструкции-выражении, соблюдая при этом их естественный порядок следования. Таким образом, вставка манипулятора ***endl*** в поток ***cout*** следом за строковым литералом возможна благодаря лишь тому, что возвращаемым значением операторной функции ***operator<<()*** является ссылка на ***ostream***, что и позволяет применить к ней следующий оператор вывода.

```
// Программирование в стиле C++ - четвертый шаг

#include <iostream>

int main()
{
    using namespace std;
    cout << "Hello, world!" << endl;
    return 0;
}
```

Примечание.

Одной из фундаментальных концепций, принятых при построении стандартной библиотеки C++, является пространство имен – область видимости (или действия).

В программе на C имеется единая глобальная область действия, что неизбежно порождает конфликты имен. С появлением пространства имен глобальная область действия стала всего лишь еще одним пространством имен. Особенность глобального пространства имен в C++ состоит только в том, что его имя не обязательно явно квалифицировать. Квалификация имени только с помощью унарного оператора разрешения области видимости **::** означает, что это имя принадлежит глобальному пространству имен, а не тому, в котором такое же имя было повторно объявлено. Классы и обычные локальные области видимости тоже являются пространствами имен. Класс – это тип, определенный именованной локальной областью видимости, которая описывает, как создаются и используются объекты данного типа.

Пространство имен является механизмом определения области действия, обобщающего глобальные объявления в C и C++. Такие области действия разрешается именовать, а к их членам можно обращаться с помощью обычного для членов класса синтаксиса. Имена, объявленные внутри пространств имен, не конфликтуют ни с глобальными именами, ни с именами из других пространств имен. Наряду с пространством имен ***std***, в котором определены средства стандартной библиотеки C++, пользователь вправе использовать и свои собственные пространства имен, поместив в них объявления необходимых ему средств.

С помощью ***using-объявлений*** имена из пространств имен можно добавлять к локальной области видимости, а ***using-директивы*** делают доступными имена из пространств имен в той области видимости, в которой они были указаны.

Иллюстрация механизма доступа к пространству имен ***std*** в локальной области видимости функции ***main*****()** основана на использовании локальной ***using-директивы***.