Модульное программирование

Модули

Модульность в языке C++ поддерживается с помощью **директив** препроцессора, **пространств** имен, **классов** памяти и раздельной компиляции.

Модуль – это независимый блок, код которого физически и логически отделен от кода других модулей.

Модуль содержит **данные** и **функции** их обработки.

Модульная организация программы позволяет оформлять **логически связанные модули** в отдельные файлы.

Заголовочные файлы

- Практически любая программа начинается с директив препроцессору
  - **#include**
    - указывает компилятору включить содержимое файла с расширением *.h (заголовочный файл) в программу;

Заголовочные файлы содержат определения, используемые компилятором для операций различных типов.

Язык позволяет создавать свои заголовочные файлы и подключать их к проекту.

Функции

«Функции используются для наведения порядка в хаосе алгоритмов…» Б. Страуструп

Функции

- Часто встречается ситуация, когда одно и то же действие необходимо выполнить в разных местах программы;
  - с разными наборами данных.
- Нецелесообразно в каждом таком месте приводить одну и ту же последовательность инструкций
  - нужно уметь один раз оформлять такие фрагменты специальным образом, а пользоваться ими многократно;

Для того чтобы использовать один и тот же код:

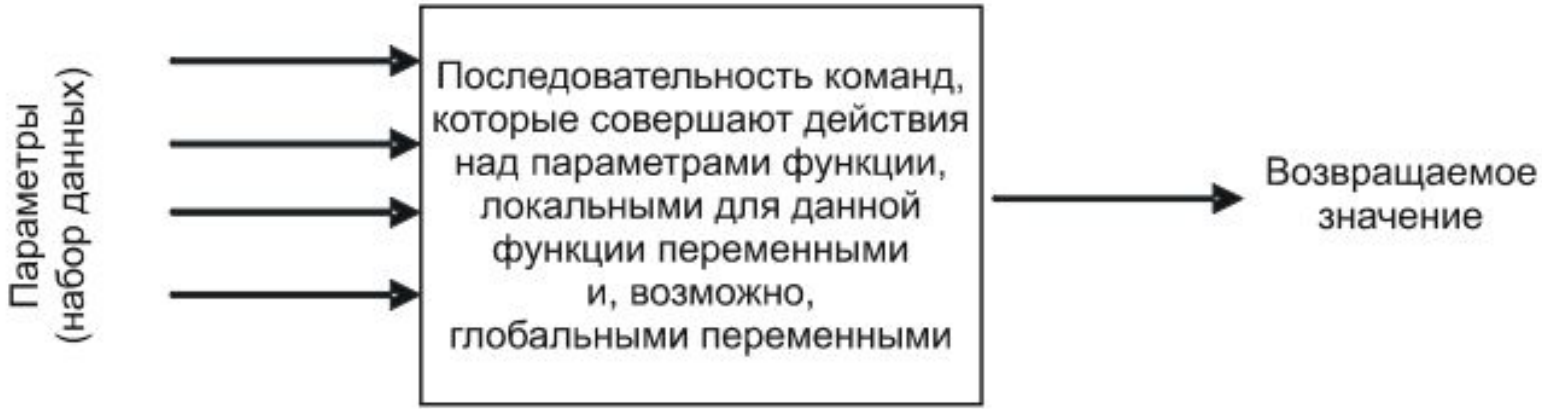
1. Сформировать исходные данные (параметры)
2. Передать управление на начало фрагмента (вызвать его).
3. По окончании выполнения фрагмента получить управление и результат работы назад

Функции

- **Функции** — это небольшие блоки(участки кода), из которых строится программа.
  - дают возможность заключить часть кода в черный ящик, который в дальнейшем можно использовать многократно;
- Функции:
  - принимают параметры,
  - выполняют последовательность инструкций
    - (называемую телом функции)
  - формируют результат,
  - возвращают управление.

Результат, формируемый функцией, представляет собой единственную (в простейшем случае скалярную) величину

Функции


Параметры
(набор данных)
Последовательность команд,
которые совершают действия
над параметрами функции,
локальными для данной
функции переменными
и, возможно,
глобальными переменными
Возвращаемое
значение

Функции

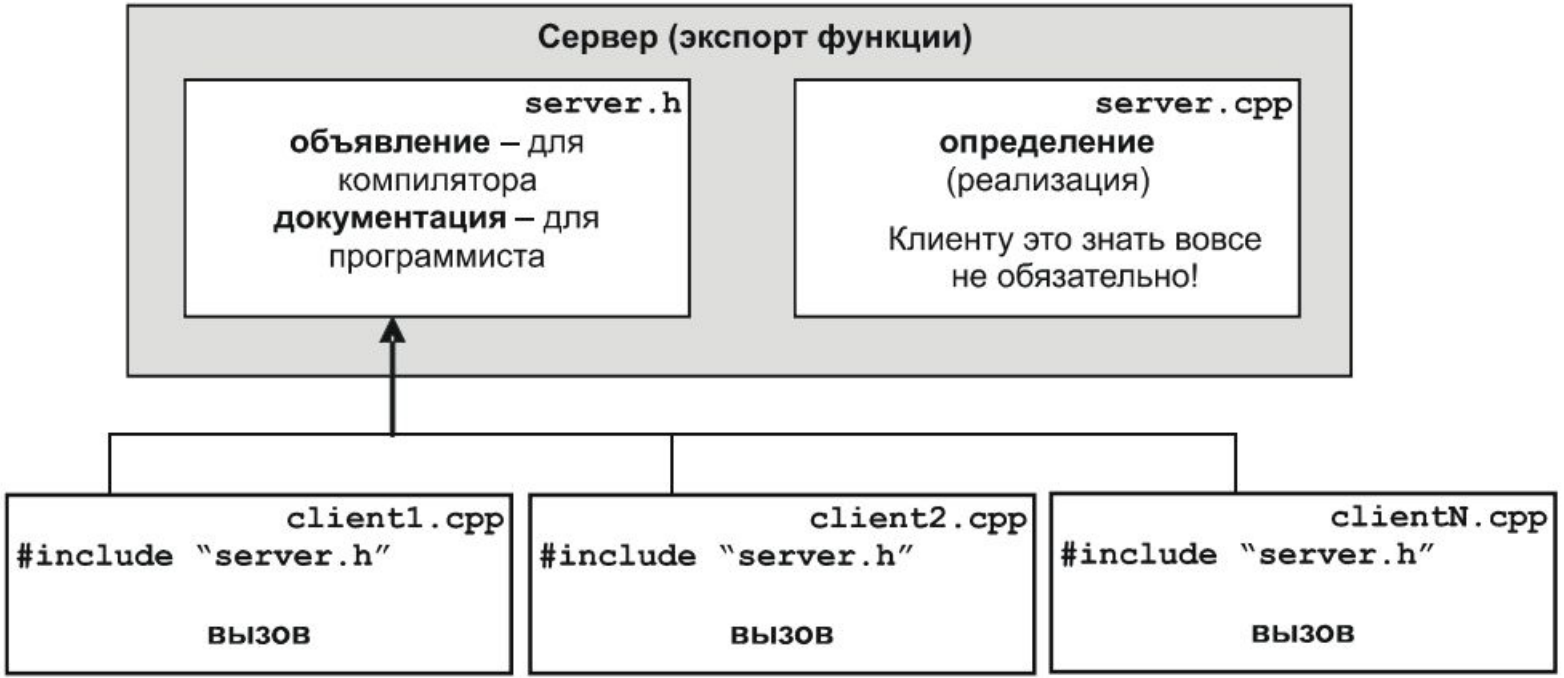
Использование функций позволяет:

- не дублировать многократно код, который выполняет одни и те же действия с разными наборами данных;
- использовать посредством функций чужой код (в частности, возможности стандартной библиотеки);
- улучшать структуру программы (путем выделения в отдельную сущность некоторого законченного действия);
- уменьшать сложность восприятия больших программ;

Функции

- Для использования функций нужно вспомнить про такие понятия, как объявление и определение;
- Как и в случае переменных, на момент использования (вызова) функции компилятор должен знать ее свойства;
- Обычно с функциями связаны три понятия:
  - объявление;
  - определение;
  - вызов.
- Согласно правилам структурного программирования модуль предоставляется в двух частях: файл реализации (.cpp), где находится тело функции, и файл интерфейса (.h), в котором описываются свойства экспортируемой функции

Функции


Сервер (экспорт функции)
server.h
объявление – для компилятора
документация – для программиста
server.cpp
определение
(реализация)
Клиенту это знать вовсе не обязательно!
client1.cpp
#include "server.h"
вызов
client2.cpp
#include "server.h"
вызов
clientN.cpp
#include "server.h"
вызов

Объявление (прототип) функции

Объявление функции — это предварительное описание, которое извещает компилятор о типе возвращаемого значения, количестве и типах передаваемых аргументов и других свойствах функции.

Используя **прототип**, компилятор может выполнить контроль числа аргументов и проверить соответствие их типов при вызове функции.

Синтаксис:

```
[спецификатор] [тип] имя_функции( [список_аргументов] );
```

Объявление (прототип) функции

Примеры:

```
char MyFunc();  //функция возвращает значение типа char
char* MyFunc(); //функция возвращает значение типа указатель на char
void MyFunc();  //функция не возвращает никакого значения
MyFunc();       //возвращаемый тип - int по умолчанию.
          //Замечание: некоторые компиляторы (VC 2005) такое умолчание
                       не поддерживают!
int MyFunc();   //то же самое, что и в предыдущем объявлении. Явное
                  указание типа возвращаемого значения является
                  предпочтительным!
```

Определение функции (реализация)

Определение функции включает те же поля, что и прототип функции, плюс тело функции, заключенное в фигурные скобки.

Тело функции — это инструкции, выполняемые при ее вызове.

```
double MyFunc(double x) //список формальных параметров (задает
                          количество, типы и формальные имена).
                        С этими формальными именами компилятор будет
                        ассоциировать те фактические значения, которые
                        переданы в стеке при вызове
{//тело функции содержит последовательность инструкций, которые будут
   осуществлять действия над фактическими значениями, переданными в стеке
      const double A=5.5, B=6.6, C=7.7;
      double result = A*x*x + B*x + C; //вычисление результата
      return result; //формирование возвращаемого значения
}
```

Определение функции (реализация)

Замечание 1. Компилятор ассоциирует имена формальных аргументов со значениями фактических в стеке. А программист пользуется этими именами как локальными переменными, область видимости и время жизни которых ограничивается фигурными скобками тела функции.

Замечание 2. Имена формальных аргументов при определении функции вовсе не обязательно должны быть такими же, как соответствующие имена в списке аргументов при вызове и тем более при объявлении.

Замечание 3. Недопустимо в теле одной функции определять другую.

Вызов функции

Для эффективного и безопасного вызова функции, компилятор должен обеспечить:

- **связь по управлению**:
  - передачу управления на начало функции и возврат управления на следующую за вызовом функции инструкцию;
  - сохранение и восстановление контекста вызывающей части;
- **связь по данным** — механизм, который позволяет передавать данные из вызывающей функции в вызываемую и возвращать результат работы функции в вызывающую функцию.

Функции. Специфика

- при формировании параметров компилятор вставляет в код вызывающей функции последовательность машинных команд, которые, в свою очередь, помещают в стек копии переменных, перечисленных в списке аргументов справа налево.
- если программист в объявлении (и определении) описал функцию как возвращающую значение, то в теле этой функции должна быть инструкция:
  - **`return`** `<выражение>;`
    - вычисляется выражение;
    - формируется возвращаемое значение, при необходимости компилятор приводит его к требуемому типу;
    - управление передается на закрывающую фигурную скобку функции.

Функции. Специфика

- если функция ничего не возвращает (тип возвращаемого значения void), то можно использовать инструкцию:
  - **return;** //без <выражения>
- инструкция return в соответствующей форме может встречаться внутри функции много раз и в любом месте
  - как только возникает необходимость возврата.
    - При этом прерывается выполнение функции, а управление передается на закрывающую фигурную скобку;

```
void f()
{
        …

        if(условие) return;

        …
}
```