Структура на C/C++ програма

Всяка програма написана на даден програмен език има определена структура. Програма, написана на Паскал има следната структура:

```
program Име_На_Програмата;
      < декларации: >
 const
 type     { Редът  не е от значение }
 var      <процедури и функции >
begin
      <оператори> { Тяло на главната  програма }
end.      { Край на програмата }
```

За програмният език С структурата на програмата има следният обобщен ви:

```
   < Команди на предпроцесора >
   < Дефиниции на типове  >
   < Прототипи на функции > Редът не е задължителен
   < Променливи >
   < Функции    >
```

Функциите, от своя страна, имат следната структура:

```
   <тип_   на_функцията>   <име_на_функцията>   (<списък   с   формални
параметри>)
   {
        <локални декларации>
        <тяло на функцията>
        return <израз>;
   }
```

където:

- **<тип_ на_функцията>** е произволен тип без масив и функционален.
- **<име_на_функцията>** е идентификатор, зададен от програмиста;
- **<списък с формални параметри>** - параметрите се разделят със запетая, името на всеки параметър се предшества от тип. Списъкът с параметри може да бъде и празен, т.е. се задава **<тип> <име>().**

Блокът може да съдържа един или повече оператора **return <израз>;**

- **<израз>** е произволен израз (в частност променлива) от типа на функцията, или съвместим с него;

Една от всички декларирани функции трябва да бъде наречена **main**. Тя ще съдържа тялото на главната програма, която ще се изпълни първа след стартиране на програмата на С и на свой ред ще осъществи обръщения към други функции. Програмата на С се състои от набор от функции, но някои от тях са от тип **void**, т.е. не връщат стойност и в този смисъл те са аналогични на процедурите в Паскал. Освен това (за разлика от Паскал), програмистът може да игнорира стойностите, върнати от функцията.

Повечето компилатори позволяват автоматично да бъде създавана структурата на конзолни приложения. Те са съобразени с идеологията на програмният език заложен от Керниган и Ритчи, но отчитат и спецификата на отделните компилатори. Основно разликите са главно в първоначално включените заглавни файлове (файлове с разширение ***.h** или ***.hpp**, които се описват в програмата посредством директивата **#include** и съдържат декларации на функции и константи). Например при MVS2010 се използва **stdafx.h**, при Borland C++ Builder се използва **vcl.h**, при DevC++ **iostream.h**. Въпреки това програмата остава с идентична структура и преминаването от един компилатор към друг не представлява особен проблем.

Структура на автоматично генериран проект от Microsoft Visual Studio е:

```
// first_main.cpp: Defines the entry point for the console application.
#include <stdafx.h>
// област за предпроцесорни директиви
// деклариране на глобални променливи, макроси и константи
void main(int argc, char * argv[])
{
  // област за програмен код
}
```

На първи ред се говори за точка на вход при конзолно приложение. Това означава, че програмата може да бъде стартирана от командния ред на Windows с това име (например **first_main.cpp**). Ред 1 представлява коментар. Коментара започват със символа **//** и се простира до края на реда. На ред 2 се подключва заглавният файл **"stdafx.h"**. Този файл прилича на контейнер и съдържа основни предпроцесорни директиви необходими за компилатора при създаването на конзолни приложения. Също така може да бъдат зададени при необходимост и други програмни единици от програмиста.

Съдържанието на този файл генериран от Microsoft Visual Studio 2005 е:

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once

#ifndef _WIN32_WINNT  // Allow use of features specific to Windows XP or
later.
#define _WIN32_WINNT 0x0501 // Change this to the appropriate value to target
other versions of Windows.
#endif

#include <stdio.h>
#include <tchar.h>

// TODO: reference additional headers your program requires here
```

#include – е предпроцесорна директива, т.е. команда за предпроцесора. Командата започва с **#** което означава, че е обръщение към предпроцесора. Предпроцесорните директиви могат да се записват и след записа **#include "stdafx.h"** до началото на главната функция. Този начин на включване на библиотеки е стандартен, докато използването на **"stdafx.h"** за деклариране е допълнително включен заглавен файл само за MVS. От ред 5 до ред 8 е обявена главната функция **main**. На ред 5 е дадена дефиницията на главната функция, която се състои от тип на върнатия резултат (в дадения пример това е **void**), името на функцията (**main**), а в кръглите скоби се задават параметрите на функцията). **void** е тип на данни който несъхранява никаква информация. Между фигурните скоби се записва основният програмен код, нарича се още **„тяло на функцията".**

Генерираният програмен код за конзолно приложение от C++ Builder се отличава от този на Microsoft и има следната структура:

```
#include <vcl.h>
#pragma hdrstop

//---------------------------------------------------------------------------

#pragma argsused
int main(int argc, char* argv[])
{
        return 0;
}
//---------------------------------------------------------------------------
```

Виждате че тук главната функция не връща резултат от тип **void** а от тип **int**. Това ни казва, че главната функция ще върне като резултат някакво цяло число,в нашия случай 0. Запомнете, че ако

главната функция е декларирана като **int main(…)** в телото на функцията трябва да се съдържа кода **return <върната стойност>;**. Това важи за всички подпрограми чийто тип на резултат не е **void**.

На ред 2 подключена библиотека **vcl.h**. Тази библиотека се включва автоматично, затова при нейното евентуално изтриване е възможно програмата да не работи. С други думи интегрираната среда автоматично създава структурата на най-простото работещо конзолно приложение за съответния компилатор. Новото в синтаксиса е предпроцесорната директива **#pragma argsused** която указва на компилатора, че програмата ще използва команди предавани и като параметри.

Общата структура на едно С приложение ще демонстрираме с код автоматично генериран от компилатора DevC++ и допълнена елементи показващи основните програмни единици .

```
// Област за заглавни файлове и предпроцесорни директиви
#include <iostream>
#include <math.h>

// област за деклариране на константите и макросите
#define PI 3.14        // деклариране на константа
#define S( R ) PI*R*R // деклариране на макрос

// Област за деклариране на прототипи на функции
float Min(float, float);

// Област за деклариране на глобални променливи и константи
const int RecordCount = 5; // Дефиниране на константа
float A, B; // Деклариране на локални променливи

using namespace std;
// Главна програма
int main(int argc, char* argv[])
{
 // деклариране на локални променливи
  float C; //
  // Програмен код
  setlocale(LC_ALL,"Bulgarian"); //  Извикване на стандартна функция
  A = 3; // Инициализация на променливата A
  B = 4; // Инициализация на променливата B
  C = Min(A,B); // Извикване на потребителска функция
  cout << "По-малкото число от " << A << " и " << B << " е " << C << endl;
  cout << "sin( C ) =  " << sin(C) << endl;
  return 0; // Върнат резултат
}

// Дефиниране на подпрограми
float Min(float a, float b ) {
   float result; // деклариране на локална променлива
   // код на подпрограмата
   if ( a < b) result = a;
   else result = b;
   return  result;  // Върнат резултат
}
```

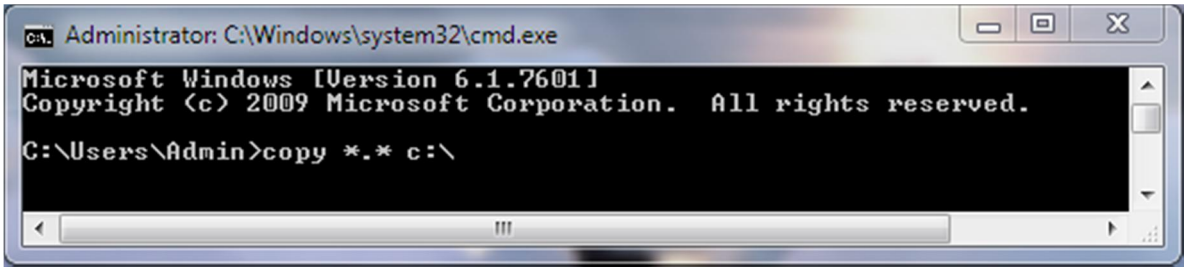
В програмният език C главната функция main може да има няколко записа. Първият синтаксис е дефинираният от Керниган и Ритчи формат прилаган в Unix платформите.

```
int main(int argc, char* argv[])
{
  return 0;
}
```

При този форма функцията **main** приема два входни параметъра **argc** и **argv**. И връща цяло число като резултат.

Параметъра **argc** е от тип **int** и указва броя на предаваните аргументи при стартиране на програмата от конзолата. Вторият параметър **argv** е масив от указатели към масив от низове. Той съдържа стойностите на придаваните параметри на програмата от командния ред. Например при изпълнение на командата за копиране на файлове от фигурата по-долу стойността на argc ще бъде 3. Що се отнася до параметъра argv, то той ще има следните стойности:

argv[0] = "C:\Users\Admin\copy", argv[1] = "*.*" и argv[2] = "c:\".



Както се вижда параметърът с индекс 0 е името на стартираното приложение. Останалите елементи са текстовете след името на програмата(командата) разделени с интервал или запетая.

Най-разширеният формат на главната функция е следният:

```
int main(int argc, char* argv[],char** envp)
{
  return 0;
}
```

Той се отличава от предходният синтаксис само по наличието на параметъра **envp**. Този параметър е указател към указател от тип char и дава информация за средата (environment), в която се изпълнява програмата. Този параметър в общия случай не задължителен.

Като цяло функцията main може да се декларира и без параметри, като най-често използваните синтаксиси са:

```
int main()
{
  return 0;
}
```

```
void main()
{
  return;
}
```

Да се върнем към примера показан по-горе. В него променливите A и B са декларирани като глобални променливи, но при желание те може да се декларират и вътре в главната програма main така както е декларирана променливата C. В много случаи такъв подход е за предпочитане, тъй като отстранява възможността за пряко модифициране на глобалните променливи от различните функции, което нарушава яснотата и четимостта на програмата и увеличава вероятността за грешки.

Може би коментарите поставени в кода на програмата ще ви се сторят странни. Това се дължи на използваните термини. Например под **декларация на променлива (variable declaration)** се разбира асоцииране на името на променлива със съответния тип данни (data type). Това само по себе си не създава променливата. Ако в декларацията се дават и стойностите, с които името се свързва, то декларацията се нарича **дефиниция**.

Идентификаторът е името на променлива или подпрограма и представлява последователност от букви и числа, като се започва с буква или символа'_', която задава името на област от паметта в която се съхраняват данни от даден тип или код на подпрограма.

Подпрограмите са такива функционално обособени програмни блокове(модули) позволяващи многократното им използване от главната програма. Те биват **функции** и **процедури** и ще бъдат разгледани по-подробно в последствие.

Предпроцесор (макропроцесор) се нарича програма, която заменя една входна последователност от символи с друга. Предпроцесорите са надстройки над езиците за програмиране като увеличават функционалните им възможности. В език C директивите на предпроцесора са оформени като отделни низове от програмата и започват със символа **"#"**, като обезпечават замяната на срещащите се в текста идентификатори с текстови низове.

Езика за програмиране C/C++ е регистровозависим. Това ще рече, че се прави разлика между големи и малки символи за разлика от езика Паскал. Например:

`Return 0;` – **е грешен запис**;

`return 0;` – **правилен запис**