Ćwiczenie nr 5

Temat: Podział aplikacji na moduły. Własne pliki nagłówkowe - dyrektywy preprocesora.

Zagadnienia:
- Dyrektywy preprocesora.
- Aplikacja złożona z wielu plików źródłowych – modułów.
- Poprawne pliki nagłówkowe.

1. Preprocesor

Preprocesor to program, którego zadaniem jest przygotowanie właściwego kodu źródłowego dla kompilatora. Programista może za pomocą dyrektyw (poleceń, zaleceń) preprocesora decydować jaki kod/fragment kodu ma być podłączany, kompilowany, przetwarzany w zależności od spełnionych określonych warunków.

2. Dyrektywy preprocesora

Wybrane, najczęściej wykorzystywane dyrektywy:

#include – dyrektywa włączająca tekst innego pliku źródłowego do pliku, w którym jest wywołana. Każda z dyrektyw include musi być umieszczona w osobnej linii tekstu (nie mogą znajdować się w jednej linii).

Przykłady:

```
#include <stdio.h>
```
- włączenie pliku nagłówkowego, plik będzie poszukiwany w standardowych katalogach środowiska programowania przeznaczonych do przechowywania plików nagłówkowych.

```
#include "moj_plik.h"
```
– włączenie pliku nagłówkowego, w którym plik będzie poszukiwany w katalogu bieżącym tworzonego programu.

```
#include "c:\stefan\include\moj_plik.h"
```
– włączenie pliku nagłówkowego z podaniem pełnej bezwzględnej ścieżki dostępu do pliku na dysku (metoda niepraktyczna).

#define - definiuje stałe, etykiety i pseudo-funkcje (makroinstrukcje).

Przykłady:

```
#define  PI  3.14
```
- wystąpienie w kodzia słowa PI zostanie zastąpione tekstem 3.14

```
#define DODAJ(a,b) ((2*a)+(2*b))
...
Y = DODAJ(2,3)
```
- makroinstrukcja (pseudo-funkcja). Wystąpienie DODAJ(2,3) zostanie zastąpione wyrażeniem (2*2)+(2*3)

```
#define  MOJA_ETYKIETA
```
- „pusta" definicja makro/etykiety. Zdefiniowana etykieta najczęściej służy sprawdzeniu czy dany fragment kodu był wcześniej analizowany przez preprocesor.

#undef - usuwa definicje stałej lub makra

Przykład:

```
#undef  PI
```
- usunięcie definicji stałej PI.

#if - dyrektywy kompilacji warunkowej

#elif - działa podobnie jak „else if" w języku C

#else - działa podobnie jak „else" w języku C

#endif - oznacza koniec bloku kompilacji warunkowej

Przykład:

```
#if OPCJA == 1                  /* jeżeli stala OPCJA = 1 */
  printf (" Opcja = 1 ");       /* drukuj tekst 1 */
#elif OPCJA == 2                /* jeżeli stala OPCJA = 2 */
  printf (" Opcja = 2 ");       /* drukuj tekst 2 */
#else                           /* jeżeli inna wartosc */
  printf (" Opcja nieznana "); /* drukuj tekst 3 */
#endif
```

#ifdef – kompilacja warunkowa sprawdzająca czy zdefiniowano makro, działa tak samo jak **#if defined(...)**

Przykład:

```
#ifdef MYMODULE                    /* jeżeli zdefiniowano makro MYMODULE */

  printf (" MYMODULE już JEST ! "); /* kod kompilowany tylko gdy jest MYMODULE */

#endif
```

#ifndef - kompilacja warunkowa sprawdzająca czy nie zdefiniowano makra, działa tak samo jak **#if !defined(...)**

```
#ifndef MYMODULE                  /* jeżeli nie zdefiniowano makro MYMODULE */

  printf (" MYMODULE nie ma ! "); /* kod kompilowany gdy nie ma MYMODULE */

#endif
```

3. Projekt aplikacji w środowisku Dev-C++ złożony z kilku plików źródłowych

W skład całego projektu wchodzi pięć plików źródłowych:
- program_matematyczny.c (program główny – main),
- math2.c (biblioteka funkcji matematycznych),
- math2.h (deklaracje funkcji zdefiniowanych w module math2.c),
- tools.c (biblioteka funkcji pomocniczych),
- tools.h (deklaracje funkcji zdefiniowanych w module tools.c).

Aby utworzyć projekt należy:
- z menu górnego wybrać: Plik → Nowy → Projekt,
- wybrać z listy opcję Empty Project (pusty projekt),
- nadać nazwę projektu: program_matematyczny,
- dodać do projektu pliki źródłowe: program_matematyczny.c, math2.c, tools.c.

4. Poprawnie zdefiniowany plik nagłówkowy – wykorzystanie dyrektyw #ifndef, #define

Plik nagłówkowy wykorzystywany w projekcie może być wykorzystany wielokrotnie (włączony do kodu dyrektywą include), np. kilka różnych plików projektu (w tym inne pliki nagłówkowe) mogą wymagać włączenia deklaracji zawartych w pliku. Jednocześnie niedopuszczalne jest aby deklaracje funkcji, zmiennych, stałych były wykonywane wielokrotnie w jednym kodzie programu. W celu zabezpieczenia przed wielokrotnym kompilowaniem tych samych deklaracji należy za pomocą dyrektywy warunkowej dopuścić do kompilacji tylko raz. Poprawnie zdefiniowany plik nagłówkowy przedstawia przykład 1. W tym przykładzie zdefiniowane makro

_TOOLS_H_ jest jednoznaczną informacją dla preprocesora że fragment kodu został dołączony wcześniej do kodu programu. Po pierwszej definicji **#define _TOOLS_H_** każde kolejne włączenie zawartości pliku nagłówkowego nie uwzględni zabezpieczonego fragmentu kodu (ten fragment nie będzie kompilowany).

Przykład 1. Plik nagłówkowy tools.h

```
#ifndef _TOOLS_H_

  /*
       jeżeli nie zdefiniowane makro _TOOLS_H_ kompiluj poniższy kod,
       ten fragment zostanie uwzględniony tylko raz bez względu na to
       ile razy plik będzie włączany w projekcie
  */

  #define _TOOLS_H_ /* definiuj makro */

  /* prototypy (deklaracje) funkcji */
  void drukuj_menu();
  void drukuj_wynik(float y);

#endif
```

Definicje funkcji należy umieścić w oddzielnym pliku źródłowym (Przykład 2. tools.c) i podłączyć do projektu. Plik ten zostanie początkowo skompilowany oddzielnie do pliku binarnego tools.o i będzie stanowił moduł aplikacji.

Przykład 2. Plik źródłowy zawierający definicje funkcji - tools.c

```
#include <stdio.h>

void drukuj_menu()
{
 printf("  Program matematyczny ver. 1.0 \n\r");
 printf("  '1' - Obliczenia wartosci funkcji y = 3x^2 + 2x - 10  \n");
 printf("  '2' - Obliczenia wartosci funkcji y = sin(x) + 2* cos(x)  \n");
 printf("  '3' - Obliczenia wartosci funkcji y = cos(x) + 2* sin(x)  \n");
 printf("  'q' - Wyjscie   \n\n");
}

void drukuj_wynik(float y)
{
 printf(" Wynik y = %5.2f \n\n", y);
 printf(" ------------------------------------------------------ \n\n" );
}
```