Podstawy programowania komputerów

Wykład 5: „Funkcje i makroinstrukcje w ANSI C”

swalover@ie.tu.koszalin.pl

Czym jest funkcja w C/C++

Funkcja - wydzielony fragment kodu źródłowego, spełniający określone zadania.

- pełnią tę samą rolę co funkcję, podprogramy i procedury w innych językach
- pozwalają uniknąć powtarzania tych samych fragmentów kodu
- stosowanie funkcji zwiększa modularność programu

Funkcję często traktowane są jako „czarne skrzynki”.

Funkcje użytkownika

Identyfikator funkcji występuje jako:

- prototyp funkcji

```
# include cosik.h
void Do_do (void);
```

- wywołanie funkcji

```
int main (void)
     {...
      Do_do( );
        ...}
```

- definicja funkcji.

```
void Do_do (void)
     {//opis funkcji
      }
```

Definicja funkcji

Funkcje należą do typów, deklaracja następuje przed wywołaniem funkcji.

```
<typ wyniku><nazwa>(<argumenty>)
    {
    <deklaracje i instrukcje>
    return <wyrażenie>;
    }
```

Instrukcja **return** powoduje wyjście z funkcji i powrót do miejsca jej wywołania. Jeżeli funkcja jest typu **void**, instrukcja ma postać **return**;

Argumenty funkcji

Deklarowanie funkcji pobierającej argumenty:

```
void Do_do (char zn, int num);
```

Argumenty formalne są zmiennymi lokalnymi

```
void ping (int x, y, z); //nieprawidłowy nagłówek
void ping (int x, int y, int z); // dobrze
void ping (x, y, z); //tak można
int x, y, z;
```

Wywołanie funkcji pobierającej argumenty:

```
Do_do ('$', 007);
Do_do (znak, numer);
```

Argumenty można przekazać poprzez zmienne

Przykład użycia funkcji

```
int a3(int a)     //a do sześcianu
{    return a*a*a;
}
int main( )
{    int x=3, y=5, z;
     z=a3(x)+a3(y);
     return 0;
}
```

Argumenty z wartościami domyślnymi

```
void RysujPunkt (int x, int y, int kolor=WHITE);
```

Przykłady wywołania:

```
RysujPunkt (5,7); //punkt w kolorze białym
RysujPunkt (7,20,RED); //w kolorze czerwonym
```

Niepoprawny zapis:

```
void RysujPunkt(int kolor=WHITE, int x, int y);
RysujPunkt (5,7); //brak trzeciego argumentu!
```

Środki standardowe

Zadanie: nadrukować pierwsze słowo powitania w lewym górnym rogu ekranu, a drugie w prawym dolnym.

```
#include <stdio.h>
#include <conio.h>
int main(){
            clrscr();
            gotoxy(5,2);
            puts("Witaj");
            gotoxy(65,24);
            puts("mistrzu");
            return 0;
            }
```

Własna funkcja

```
#include <stdio.h>
#include <conio.h>
void pisz(int x, int y, char napis[])
          {    gotoxy(x,y);
               puts(napis);   }
int main()
          {    clrscr();
               pisz(5,2,"Witaj");
               pisz(65,24,"mistrzu");
               return 0;
          }
```

Makroinstrukcje (makrodefinicje)

```
#define <nazwa> <reszta wiersza>
```

/* Każde wystąpienie identyfikatora **<nazwa>** w kodzie źródłowym preprocesor zastępuje zawartością **<reszta wiersza>** Makrodefinicja może zawierać argumenty */

```
#define DRUKUJ(a,b) cout<<(a)<<(b)
 DRUKUJ("Pole trojkonta= ", a*h/2);
```

//Zostanie zastąpione przez preprocesor instrukcją

```
 cout<<("Pole trojkonta =")<<(a*h/2);
```

Zastosowanie makrodefinicji

- Utrudnia lokalizację błędów: kompilator widzi postać rozwiniętą;
- makrodefinicje nie mogą być przeciążone, nie można stosować rekursji;
- makrodefinicję z argumentami w kodzie wyglądają podobnie do wywołania funkcji;
- należy je stosować tylko wtedy gdy jest to konieczne dla skrócenia zapisu.