Wykład VI - funkcje

niedziela, 22 kwietnia 2018 09:23

Funkcje

- blok instrukcji wykonywany, gdy zostanie wywołany z pewnego miejsca programu;
- nazywany też podprogramem, rzadziej procedurą;

Program:

```
/* program z powtarzajacymi się fragmentami */
#include <stdio.h>

int main()
{
	for(i=1; i<=5; ++i)
	{
		printf("%d", i*)i;
	}
	for(i=1; i<=5; ++i)
	{
		printf("%d", i*i*i);
	}
	for(i=1; i<=5; ++i)
	{
		printf("%d", i*i);
	}
	return 0;
}
```

Tworzenie funkcji

typ nazwa(argument1, argument2,…) instrukcja

typ - typ wartości funkcji
argument1, argument2 - argumenty funkcji, każdy oznacza zapis ***typ nazwa***
nazwa - nazwa funkcji

- Funkcja może nie mieć żadnych argumentów (między nawiasami nie ma żadnego argumentu lub jest słowo void);
- Funkcje definiuje się poza główną funkcją programu (main);
- W języku C nie można tworzyć zagnieżdżonych funkcji (funkcji wewnątrz innych funkcji).

PRZYKŁAD

```
int iloczyn(int x, int y)
{
	int iloczyn_xy;
	iloczyn_xy = x*y;
	return iloczyn_xy;
}
```

PRZYKŁAD

int iloczyn(int x, int y)
{
 int iloczyn_xy;
 iloczyn_xy = x*y;
	int(..) - nagłówek funkcji;
	int iloczyn_xy - zmienna lokalna

```
	return iloczyn_xy;
}
```

PROGRAM

Kod	Obraz na ekranie

/*wykorzystanie funkcji */
#include <stdio.h>
int dodawanie (int a, int b)
{
 int r;
 r=a+b;
 return (r);
}
 	
int main()
{
 int z;
 z = dodawanie (5, 3);
 printf("Wynik: %i", z);
 return 0;
}
	Wynik: 8

Tworzenie procedur

void nazwa(argument1,argument2,…) instrukcja

- **void** oznacza brak wartości;
- przyjęło się, że od funkcji procedura różni się tym, że nie zwraca żadnej wartości
- generalnie w C pojęcie "procedura" nie jest używane, mówi się "funkcja zwracająca void"
- gdy nie ma podanego typu danych zwracanych przez funkcję, kompilator domyślnie przyjmie typ int;

Wywoływanie funkcji

nazwa (argument1,argument2,…)

- argument1, argument2 - argumenty funkcji; każdy oznacza zapis *nazwa;*
- wywołując funkcję nie przyjmującą żadnych argumentów należy podawać jej nazwę wraz z nawiasami

Przykład:

```
void pisz_komunikat()
{
	printf("To jest komunikat\n");
}

pisz_komunikat; /* źle */
pisz_komunikat(); /*dobrze*/
```

Przypisanie zmiennej wartości, którą zwraca funkcja

zmienna = funkcja(argument1, argument2,…)

- **zmienna** - nazwa zmiennej
- **funkcja** - nazwa funkcji
- **argument1**, argument2 - argumenty funkcji; każdy oznacza zapis *nazwa*

PROGRAM

Kod	Obraz na ekranie

/*wykorzystanie fukncji*/
#include <stdio.h>
 	
int suma(int a, int b)
{
return a+b;
}
 	
int main()
{
int m = suma (4, 5);
printf("4+5=%d\n", m);
return 0;
}
	4+5=9

Zwracanie wartości

return

- instrukcja służąca do przerwania funkcji i zwrócenia jej wartości lub przerwania funkcji bez zwracania wartości;

return zwracana_wartość;

- dla funkcji

return 0;

- dla procedury

Przekazywanie parametrów

- wartości argumentów, z którymi wywoływana jest funkcja, są do niej kopiowane - argumenty są przekazywane przez wartość;
- modyfikowanie zmiennych przekazywanych do funkcji jako parametry jest możliwe z wykorzystaniem wskaźników

Funkcja rekurencyjna

- funkcja, która w swojej własnej definicji (ciele) wywołuje samą siebie.

Przykład

```
int silnia(int liczba)
{
	int sil;
	if(liczba<0) return 0; /*wywołanie jest bezsensowne, zwracamy 0 jako kod błędu*/
	if(liczba==0 || liczba==1) return 1;
	sil = liczba*silnia(liczba-1);
	return sil;
}
```

Deklaracja funkcji

Program

/*program z deklaracjami funkcji*/ #include <stdio.h> **int b (int p)**	• **int b (int p)** - deklaracja funkcji

```
int a()
{
	return b(0);
}

int b (int p)
{
	if(p==0)
		return 1;
	else
		return a();
}

int main()
{
	return b(1);
}
```

(prototyp funkcji) - funkcje mogą wywoływać tylko funkcje, które zostały wczytane przed nimi

Program

Kod	**Obraz na ekranie**

/*program z deklaracjami funkcji 1*/
#include <stdio.h>
 	
void odd(int a);
void even(int a);
 	
int main()
{
int i;
do {
printf("Wprowadz liczbe: (0 by wyjsc): "); scanf("%i", &i); odd (i);
} while (i!=0);
return 0;
}
 	
void odd (int a)
{
if((a%2)!=0) printf("Liczba nieparzysta. \n");
else even (a); return;
}
 	
void even (int a)
{
if((a%2)==0) printf("Liczba parzysta. \n");
else odd (a); return;)
}
	Wprowadz liczbe: (0 by wyjsc):
	9
	Liczba nieparzysta.
	Wprowadz liczbe: (0 by wyjsc):
	6
	Liczba parzysta.
	Wprowadz liczbe: (0 by wyjsc):
	1030
	Liczba parzysta.
	Wprowadz liczbe: (0 by wyjsc):
	0
	Liczba parzysta.