

Programowanie strukturalne

- wykład 3

dr Piotr Jastrzębski

Funkcja

Funkcje w C

Ogólna składnia funkcji

```
typ identyfikator (typ1 argument1, typ2 argument2)
{
    /* instrukcje */
}
```

procedura

```
void identyfikator (typ1 argument1, typ2 argument2)
{
    /* instrukcje */
}
```

Rekurencja

Rekurencja, zwana także rekursją (ang. recursion, z łac. recurrere, przybiec z powrotem) – odwoływanie się np. funkcji lub definicji do samej siebie.

$$F_n := \begin{cases} 0 & \text{dla } n = 0, \\ 1 & \text{dla } n = 1, \\ F_{n-1} + F_{n-2} & \text{dla } n > 1. \end{cases}$$

Pamięć w języku C

Pamięć w języku C

Program w języku C po skompilowaniu wykorzystuje trzy rodzaje pamięci:

- ▶ statyczną (globalną) - korzystają z tego zmienne statyczne i globalne.
- ▶ automatyczną - zmienne zadeklarowane wewnątrz funkcji
- ▶ dynamiczna - alkowa na stercie i może być zwolniona, gdy będzie to konieczne.

Zmienne statyczne:

```
#include <stdio.h>
#include <stdlib.h>
```

```
void foo()
{
    static int x=0;
    x++;
    printf("liczba wywolan: %d\n",x);
}
```

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

Zasięg zmiennych:

Rodzaj pamięci	Zasięg	Okres istnienia
globalna	cały plik	cały okres działania aplikacji
statyczna	funkcja, w której została zadeklarowana	cały okres działania aplikacji
automatyczna (lokalna)	funkcja, w której została zadeklarowana	czas wykonywania funkcji
dynamiczna	określony przez wskaźniki odnoszące się do tej pamięci	do momentu zwolnienia pamięci

Wskaźniki

Definicja

Wskaźnik (ang. pointer) to specjalny rodzaj zmiennej, w której zapisany jest adres w pamięci komputera.

Symbol	znaczenie	użycie
*	weź wartość x	*x
*	deklaracja wskaźnika do wartości	int *x;
&	weź adres	&x

```
#include <stdio.h>

int main (void)
{
    int liczba = 5;
    printf("Wartość zmiennej: %d\n", liczba );
    printf("Adres zmiennej: %p\n", &liczba );
    printf("Adres zmiennej: %#010x\n", &liczba );
    return 0;
}
```

Przeanalizujmy kod:

```
void idzPrawoDol(int x, int y)
{
    x=x+1;
    y=y-1;
}

int main()
{
    int x=20, y=15;
    idzPrawoDol(x,y);
    printf("Aktualna pozycja: [ %d, %d ] \n",x,y);
    return 0;
}
```

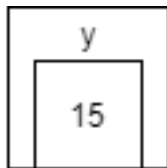
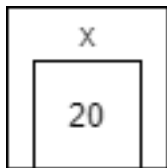
Na wyjściu otrzymujemy:

Aktualna pozycja: [20, 15]

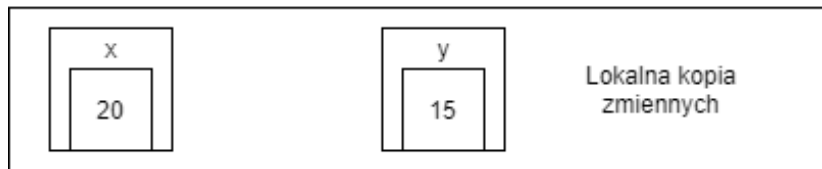
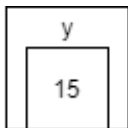
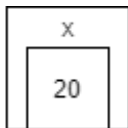
Dlaczego tak się dzieje?

W języku C argumenty przekazywane są przez wartość.

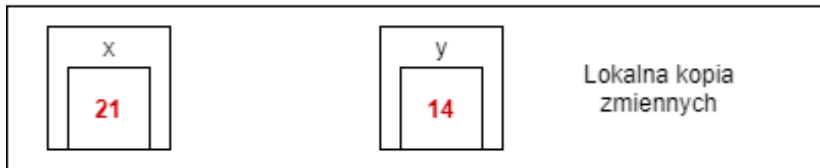
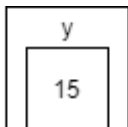
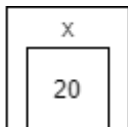
Na początku funkcji `main` mamy dwie zmienne:



Podczas wykonywania funkcji `idzPrawoDo1` następuje kopiowanie wartości do zmiennych lokalnych:



Potem zmiana zmiennych lokalnych:



Jak to naprawić? Użyjemy wskaźników.

```
void idzPrawoDol(int *x, int*y)
{
    *x=*x+1;
    *y=*y-1;
}

int main()
{
    int x=20, y=15;
    idzPrawoDol(&x,&y);
    printf("Aktualna pozycja: [ %d, %d ] \n",x,y);
    return 0;
}
```

Aktualna pozycja: [21, 14]

Ważne, że poniższe zapisy są równoważne:

```
int* x;  
int * x;  
int *x;  
int*x;
```

Zastosowania wskaźników

- ▶ tworzenie szybkiego i wydajnego kodu,
- ▶ rozwiązywanie w prosty sposób różnego typu problemów,
- ▶ obsługa dynamicznej alokacji pamięci,
- ▶ tworzenie zwięzłych wyrażeń,
- ▶ przekazywanie struktur danych bez ponoszenia kosztów w postaci narzutu,
- ▶ ochrona danych przekazywanych do funkcji jako parametry.

Referencje w C?

[https://pl.wikipedia.org/wiki/Referencja_\(informatyka\)](https://pl.wikipedia.org/wiki/Referencja_(informatyka))

System szesnastkowy

https://pl.wikipedia.org/wiki/Szesnastkowy_system_liczbowy

Wskaźnik na stałą wartość, a stały wskaźnik

Wskaźnik na stałą wartość:

```
const int *a;  
int const * a;
```

Stały wskaźnik:

```
int * const b;
```

```
int i=0;
  const int *a=&i;
  int * const b=&i;
  int const * const c=&i;
*a = 1;  /* kompilator zaprotestuje */
*b = 2;  /* ok */
*c = 3;  /* kompilator zaprotestuje */
a = b;   /* ok */
b = a;   /* kompilator zaprotestuje */
c = a;   /* kompilator zaprotestuje */
```


Funkcja malloc

```
void *malloc(size_t size);
```

Funkcja służy do dynamicznego rezerwowania miejsca w pamięci. Gdy funkcja zostanie wywołana, w przypadku sukcesu zwróci wskaźnik do nowo zarezerwowanego miejsca w pamięci; w przypadku błędu zwraca wartość NULL.

```
int num = 0;
int *pi = &num;
printf("Adres pi: %d Wartosc: %d\n",&pi, pi);
printf("Adres pi: %x Wartosc: %x\n",&pi, pi);
printf("Adres pi: %o Wartosc: %o\n",&pi, pi);
printf("Adres pi: %p Wartosc: %p\n",&pi, pi);
```

Wyłuskiwanie (dereferencja) wskaźnika

```
int num = 5;  
int *pi = &num;  
printf("%p\n", *pi);
```

Wskaźniki na funkcję

```
typ_zwracany (*nazwa_wsk)(typ1 arg1, typ2 arg2);
```

```
#include <stdio.h>

int suma (int lhs, int rhs)
{
    return lhs+rhs;
}

int main ()
{
    int (*wsk_suma)(int a, int b);
    wsk_suma = suma;
    printf("4+5=%d\n", wsk_suma(4,5));
    return 0;
}
```

Bibliografia

- ▶ Richard Reese, Wskaźniki w języku C, Wydawnictwo Helion 2014.
- ▶ <https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki>, dostęp online 15.03.2020.
- ▶ http://wazniak.mimuw.edu.pl/index.php?title=Wst%C4%99p_d_o_programowania_w_j%C4%99zyku_C/Wska%C5%BAniki, dostęp online 15.03.2020.
- ▶ https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki_-_wi%C4%99cej, dostęp online 15.03.2020.