Wskaźniki

Wskaźnik — zmienna przechowująca adres pamięci

Wskaźniki

Wskaźnik — zmienna przechowująca adres pamięci

```
int main() {
  int x = 5;
  int* ptr_x = NULL; //tworzymy wskaznik
  ptr_x = &x;  //i przypisujemy do niego adres x
  printf("%p", ptr_x); //wypisujemy wskaznik
  printf("%lu, sizeof(ptr_x));
}
```

Dlaczego potrzebujemy wskaźników (slajd z wykładu WDP w Języku C, ©MPi)

Po co są wskaźniki?

Dlaczego potrzebujemy wskaźników (slajd z wykładu WDP w Języku C, ©MPi)

Po co są wskaźniki?

Współdzielenie struktur danych przez rózne fragmenty kodu (a kopiowanie danych moze być nieefektywne).

Dlaczego potrzebujemy wskaźników (slajd z wykładu WDP w Języku C, ©MPi)

Po co są wskaźniki?

Współdzielenie struktur danych przez rózne fragmenty kodu (a kopiowanie danych moze być nieefektywne).

Budowanie złozonych struktur danych (np. drzew czy grafów) poprzez dynamiczne łączenie ich fragmentów.

Podstawowe operacje na wskaźnikach

```
int main() {
  int x = 5;
  int* ptr_x = &x;
  int y = *ptr_x;
  printf("%d\n", *ptr_x); //deferencja
  int* ptr = ptr_x + 5;
  ptr++; // += 1 tez dziala
  printf("%p %p\n", ptr, ptr_x);
  long long int z = 10;
  long long int* ptr_z = &z;
  printf("%p %p\n", ptr_x, ptr_x + 1 );
  printf("%p %p\n", ptr_z, ptr_z + 1 );
}
```

Wskaźnik na strukturę

```
typedef struct struktura
{
  int a, b;
} Struktura;

int main() {
  Struktura* s;
  s->a = 50;
 (*s).a = 50;
}
```

Przekazywanie przez ref/wartosc

```
void zwieksz_x_zle(int arg) {
  arg++; //arg jest lokalne!
}

void zwieksz_x(int *arg) {
  (*arg)++; //zmieniamy to na co wskazuje arg
}

int main() {
  int x = 0;
  zwieksz_x_zle(x);
  printf("%d\n", x);
  zwieksz_x(&x);
  printf("%d\n", x);
}
```

Sterta i tablice dynamiczne

Ostatnio nauczyliśmy się, że nie można alokować dynamicznie dużych tablic przez `int tab[n]`

Sterta i tablice dynamiczne

Ostatnio nauczyliśmy się, że nie można alokować dynamicznie dużych tablic przez `int tab[n]`
Jak zatem alokować duże tablice dynamiczne?

Sterta i tablice dynamiczne

Ostatnio nauczyliśmy się, że nie można alokować dynamicznie dużych tablic przez `int tab[n]`
Jak zatem alokować duże tablice dynamiczne?

```
int main() {
  int size = 10000000;
  // int tab[size]; //poprzednio to nie dzialalo
  int* tab = (int*)malloc(size*sizeof(int));
  for (int i=0; i<size; ++i) tab[i] = 0;

  int* ptr = tab; int* end = tab+size;
  while(ptr++ != end) *ptr = 0;

}
```

Alokujemy `size*sizeof(int)` bajtów `tab` jest wskaznikiem na początek zaalokowanych danych.

Sterta i tablice dynamiczne

```
int main() {
  int size = 10000000;
  int* tab = (int*)malloc(size*sizeof(int));
  *(tab+10) = 4; //tab+10 --- wskaznik na 11 element,
  tab[10] = 5; // to samo co wyżej!

  int tab2[10];
  tab2[0] = 10;
  printf("%d\n", *tab2);
}
```

Sterta i tablice dynamiczne

```
int main() {
  int size = 10000000;
  int* tab = (int*)malloc(size*sizeof(int));
  *(tab+10) = 4; //tab+10 --- wskaznik na 11 element,
  tab[10] = 5; // to samo co wyżej!

  int tab2[10];
  tab2[0] = 10;
  printf("%d\n", *tab2);
}
```

wartość tab2 to adres pamieci początku tablicy;

malloc/realloc/free

```
void do_something() {
  size_t size = 10;
  int* tab = (int*)malloc(size);
  if(tab == NULL)
    printf("Nie udalo sie przydzielic pamieci\n");
  else
    tab = (int*)realloc(tab, 1000);
  free(tab);
}

int main() {
  do_something();
}
```

malloc/realloc/free

```
void do_something() {
  size_t size = 10;
  int* tab = (int*)malloc(size);
  if(tab == NULL)
    printf("Nie udalo sie przydzielic pamieci\n");
  else
    tab = (int*)realloc(tab, 1000);
  free(tab);
}

int main() {
  do_something();
}
```

Należy pamietać o zwalnianiu pamięci!
Wyciek pamięci (ang. memory leak)

Przykład

```
int wypelnij(int* t, int rozmiar) {
  for (int i=0; i<rozmiar; ++i)
    t[i] = i*i;
}

int main() {
  int tab[4];
  wypelnij(tab, 4);
}
```

Tablice dwuwymiarowe

```
int main() {
  int n = 5, m = 10;

  int** tab = (int**)malloc(n * sizeof(int*));
  for (int i=0; i<m; i++)
    tab[i] = (int*)malloc(m * sizeof(int));

  tab[2][3] = 10;
  for (int i=0; i<m; i++)
    free(tab[i]);
  free(tab);

}
```