Wskaźniki / adresy

Wskaźnik jest zmienną, która zawiera adres (wskazanie) innej zmiennej lub adres dowolnego obszaru w pamięci komputera, (np. może być to adres obszaru danych lub adres kodu programu).

Ogólna postać definicji wskaźnika wygląda następująco:

typ_danych * identyfikator wskaźnika ;

Najczęściej używane są wskaźniki „zdefiniowane" zawierające adres innej zmiennej. Taki wskaźnik zawiera informację o:

- adresie zmiennej w pamięci komputera
- typie danych przechowywanych w tej zmiennej

Przykłady definicji

```
int    * wskaznik;     // wskaźnik na zmienną całkowitą
double * wsk_liczby;   // wskaźnik na zmienną rzeczywistą
char   * wsk_znaku;    // wskaźnik na pojedynczy znak
char   * tekst;        // wskaźnik na początek łańcucha znaków
                          (na pierwszy znak tego łańcucha)
```

Można również korzystać ze wskaźników „niezdefiniowanych" (anonimowych). Taki wskaźnik zawiera tylko informację o adresie początku obszaru pamięci (bez określenia typu wskazywanych danych). Definicja takiego wskaźnika ma postać:

void * identyfikator wskaźnika ;

jest to wskaźnik na „dowolny" ciąg bajtów danych.

W powyższym przykładzie widzimy, że pierwsza i trzecia pętla *for* są takie same. Zamiast kopiować fragment kodu kilka razy (co jest mało wygodne i może powodować błędy) lepszym rozwiązaniem mogłoby być wydzielenie tego fragmentu tak, by można go było wywoływać kilka razy. Tak właśnie działają funkcje.

Operatory „wskaźnikowe"

Ze wskaźnikami i adresami związane są dwa operatory:

- operator adresu (referencji) **&** zwracający adres zmiennej podanej po prawej stronie tego operatora
- operator wyłuskania (dereferencji) ***** identyfikujący obszar zmiennej wskazywanej przez wskaźnik podany po prawej stronie tego operatora.

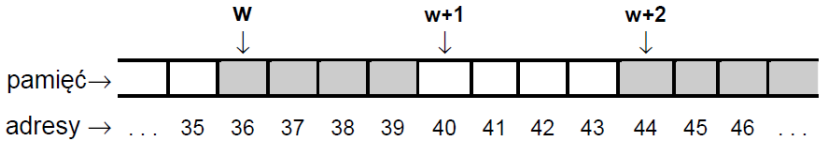
Przykład

```
int liczba ;
int *wskaźnik ;
wskaznik = &liczba;     // przypisanie zmiennej wskaźnik
                        // adresu zmiennej liczba

*wskaźnik = 10;         // przypisanie wartości 10 do zmiennej
                        // wskazywanej przez wskaźnik
                        // ( tutaj jest to równoważne liczba = 10 )
```

Arytmetyka wskaźników

Na wskaźnikach mogą być wykonywane następujące operacje:

- przypisania (=)
  wsk = wskaznik_zmiennej_lub_obszaru_pamięci ;
  (w przypadku niezgodności typów konieczne jest dokonanie konwersji typu)
- operacje porównania (==, !=, <, >, <=, >=)
  wsk_1 == wsk_2 // sprawdzenie czy zmienne zawierają te same adresy
  wsk_1 < wsk_2 // czy zmienna **wsk_1** zawiera adres mniejszy
  // od adresu zawartego w zmiennej **wsk_2**
- operacje powiększania lub pomniejszania wskaźnika (+, -, ++, --, +=, -=) o liczbę całkowitą (tylko dla wskaźników zdefiniowanych)
  - powiększenie (pomniejszenie) wskaźnika o wartość N powoduje wyznaczenie adresu przesuniętego o:
    N * sizeof(*typ_zmiennej_wskazywanej* **)**
    bajtów w kierunku rosnących (malejących) adresów.
    Np. **int *w**



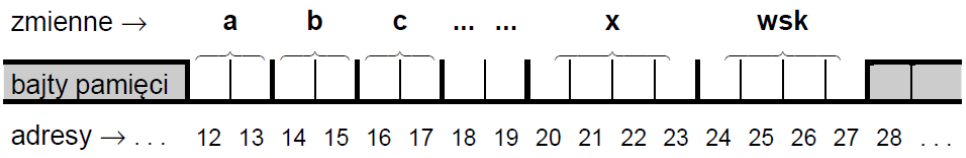
- operacje odejmowania wskaźników tego samego typu, rozumiane jako wyznaczenie „odległości” pomiędzy dwoma adresami w pamięci.
  - odległość w sensie: **N * sizeof (** *typ_elementu_wskazywanego* **)**

Przykłady zmiennych wskaźnikowych

```
int  *   wsk_liczby;  // wskaźnik na liczbę typu int

int      tab_A[10];   // 10-cio elementowa tablica liczb int
                      ( identyfikator  tab_A  jest stałą równą adresowi
                       pierwszego elementu tablicy o tej samej nazwie
                       tzn.    tab_A == &( tab_A[0] )

int  *   tab_B[10];   // 10-cio elementowa tablica wskaźników na liczby int
int  *  ( tab_C[10] ); // jak wyżej
( int *)   tab_D[10];  // jak wyżej

int     (*tab_E)[10];  // wskaźnik na 10-cio elementową tablicę liczb int
```

Dostęp do zmiennej za pomocą wskaźników - przykład

```
#include <stdio.h>
short a ;       //  typ danych short ≡ short int
short b ;       //  w systemach 32 bitowych zajmuje w pamięci 16 bitów (2 bajty)
short c ;
float x ;
short * wsk ;

 // przykładowa organizacja zajętości pamięci komputera przy w/w definicjach
```



```
int main( )        // Program ilustrujący różne sposoby zapisu wartości do zmiennej 'b' :
{                  // bezpośrenio (poprzez nazwę zmiennej) oraz pośrednio, za pomocą
                   // wskaźnika na 'b' lub na sąsiadujące zmienne 'a' , 'c', 'x'
   // Wyświetlenie adresów przydzielonych zmiennym: a, b, c, x, wsk
   printf(" \n Adres zmiennej A = %u " , (unsigned) &a ) ;          // 4203212
   printf(" \n Adres zmiennej B = %u " , (unsigned) &b ) ;          // 4203214
   printf(" \n Adres zmiennej C = %u " , (unsigned) &c ) ;          // 4203216
   printf(" \n Adres zmiennej X = %u " , (unsigned) &x ) ;          // 4203220
   printf(" \n Adres zmiennej WSK = %u " , (unsigned) &wsk ) ;      // 4203224

   a = b = c = 0;                  printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   b=10;                           printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   wsk = &b;
   *wsk = 20;                      printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   wsk = &a;
   *(wsk +1) = 30;                 printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   *(&a + 1) = 40;                 printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   *(&c – 1) = 50;                 printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   *( (short*)&x –3) = 60;         printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   *((short *)(&x –1) –1) = 70;    printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   *((short *)&wsk –5) = 80;       printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   *((short*)(&wsk –2) –1)= 90;    printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
   getchar();
}
```

Dostęp do tablic za pomocą indeksów i/lub wskaźników – przykłady

```c
#include <stdio.h>
#define ROZMIAR 10
int main(void)
{
    int tab[ROZMIAR];
    int i; // dostep do elementow tablicy za pomoca operatora indeksu
    for(i=0; i<ROZMIAR; i++)
    {
        printf("podaj element: ");
        scanf("%d", &tab[i]); // wczytanie liczby do tablicy
        tab[i] = 2*tab[i]; // przemnozenie elementu tablicy przez 2     tab[i] *= 2;
        printf("\tTab[%d] = %d\n", i, tab[i]); // wyswietlenie elementu tablicy
    }
    return 0;
}
```

```c
#include <stdio.h>
#define ROZMIAR 10
int main(void)
{
    int tab[ROZMIAR];
    int i; // dostep do elementow tablicy za pomoca indeksu i operatora wyluskania
    for(i=0; i<ROZMIAR; i++)
    {
        printf("podaj element: ");
        scanf("%d", tab+i); // &*(tab+i) == tab+i
        *(tab+i) = 2 * *(tab+i); // *(tab+i) *= 2;
        printf("\tTab[%d] = %d\n", i, *(tab+i));
    }
    return 0;
}
```

```c
#include <stdio.h>
#define ROZMIAR 10
int main(void)
{
    int tab[ROZMIAR];
    int licznik, *wsk; // dostep za pomoca wskaznika i operatora wluskania
    for(licznik=0, wsk=tab; licznik < ROZMIAR; licznik++, wsk++)
    {
        printf("podaj element: ");
        scanf("%d", wsk);
        *wsk = 2 * *wsk; // *wsk *= 2;
        printf("\tTab[%d] = %d\n", licznik, *wsk );
    }
    return 0;
}
```

```c
#include <stdio.h>
#define ROZMIAR 10
int main(void)
{
    int tab[ROZMIAR];

    int *wsk; // dostep za pomoca samych wskaznikow (bez dodatkowego licznika)
    for(wsk=tab; wsk < tab + ROZMIAR; wsk++)
    {         // wsk < &tab[ROZMIAR] - adres "konca tablicy"
        printf("podaj element: ");
        scanf("%d", wsk);
        *wsk *= 2;
        printf("\tTab[%d] = %d\n", wsk-tab , *wsk);
    }
    return 0;
}
```

Zadania

1. Przeanalizuj przykład obrazujący dostęp do zmiennej za pomocą wskaźników. Jakie wartości zmiennych *a*, *b*, *c* zostaną wypisane na konsolę przez kolejne wywołania funkcji *printf* z tymi zmiennymi? Odpowiedź uzasadnij. Weź pod uwagę adresy zmiennych użyte w tym przykładzie.
2. Przeanalizuj przykład obrazujący dostęp do tablic za pomocą indeksów i/lub wskaźników. Jakie dostrzegasz wnioski?
3. Zdefiniuj wskaźnik na zmienną typu całkowitego. Zainicjuj wskaźnik adresem zmiennej całkowitej *int x = 20*. Wypisz na ekran zawartość zmiennej za pomocą wskaźnika oraz za pomocą identyfikatora zmiennej.
4. Napisz program wczytujący tablicę dziesięcioelementową wypełnioną dowolnymi liczbami całkowitymi. Następnie, wykorzystując wskaźniki, wyświetl:
    a. wartość pierwszego elementu tablicy,
    b. wartość elementu piątego,
    c. zawartość całej tablicy.