WSKAŹNIKI / ADRESY

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

Ogólna postać definicji wskaźnika:

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** obszaru pamięci (bez określenia typu wskazywanych danych). Definicja takiego wskaźnika ma postać:

void * identyfikator wskaźnika ;

jest to wskaźnik na „dowolny" ciąg bajtów danych.

Ze wskaźnikami i adresami związane są dwa operatory:

- operator referencji **&** zwracający adres zmiennej podanej po prawej stronie tego operatora.
- operator dereferencji * identyfikujący obszar wskazywany przez wskaźnik podany po prawej stronie tego operatora.

```
int liczba ;
int *wskaźnik ;
wskaznik = &liczba;      // przypisanie zmiennej wskaźnik
                         // adresu zmiennej liczba
*wskaźnik = 10;          //  przypisanie 10 zawartości zmiennej
                         // wskazywanej przez wskaźnik
                         // tutaj 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 (==, !=, <, >, <=, >=):

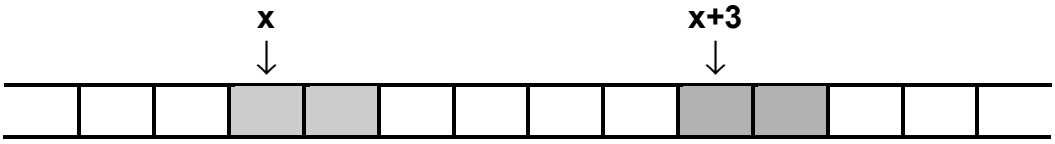
  wsk_1 == wsk_2 // sprawdzenie czy zmienne zawierają te same adresy
  wsk_1 < wsk_2 // czy zmienna **wsk_1** zawiera adres mniejszy
  // od adresu zawartego w zmiennej **wsk_2**

- 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** ∗**x**;



adresy → . . . 35 36 37 38 39 40 41 42 43 44 45 46 . . .

- operacje odejmowania wskaźników tego samego typu → wyznaczenie „odległości" pomiędzy dwoma adresami w pamięci.
  (odległości 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])

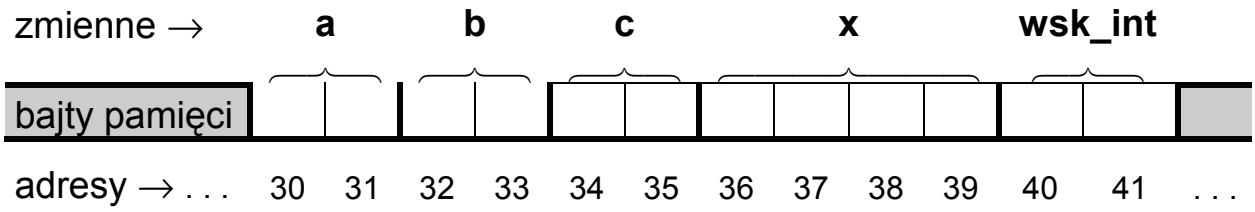
int ∗ **tab_B**[10]; // *10-cio elementowa tablica wskaźników na liczby* ***int***
int ∗ (**tab_C**[10]); // *jak wyżej*
(**int** ∗) **tab_D**[10]; // *jak wyżej*

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

PRZYKŁADY: Dostęp do zmiennych za pomocą wskaźników

```
#include <stdio.h>
int a ;
int b ;
int c ;
float x ;
int* wsk_int ;
```

// organizacja zajętości pamięci komputera przy w/w definicjach




```
void main( )                    // Różne sposoby zapisu nowej wartości do zmiennej b
{                               // za pomocą wskaźnika na b i sąsiadujące zmienne
   // Wyświeltlenie adresów zmiennych
   printf(" \n Adres zmiennej A = %u " , (unsigned) &a ) ;
   printf(" \n Adres zmiennej B = %u " , (unsigned) &b ) ;
   printf(" \n Adres zmiennej C = %u " , (unsigned) &c ) ;
   printf(" \n Adres zmiennej X = %u " , (unsigned) &x ) ;
   printf(" \n Adres zmiennej WSK_INT = %u " , (unsigned) &wsk_int ) ;

   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_int = &b;
   *wsk_int = 20;                   printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;

   wsk_int = &a;
   *(wsk_int+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 ) ;

   *( (int*)&x – 2) = 60;           printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;

   *(int*)( &x – 1) = 70;           printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;

   *((int*)&wsk_int – 4) = 80;      printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;

   *(int*) (&wsk_int – 4) = 90;     printf( " \n A=%d, B=%d, C=%d " , a, b, c ) ;
}
```

PRZYKŁADY: Dostęp do tablic za pomocą indeksów i/lub wskaźników

```
#include <stdio.h>  // Część wspólna przykładów na tej stronie
#define   ROZMIAR  10

void main(void)
{
   int tab[ ROZMIAR ];

            // wczytanie liczby do tablicy
   • • •    // przemnożenie elementu tablicy przez 2
            // wyświetlenie elementu tablicy
}
```

a)
```
int i;                  // dostęp za pomocą indeksu
for( i = 0; i < ROZMIAR; i++ )
   {
      scanf( "%d", &tab[ i ] );
      tab[ i ] = 2 * tab[ i ];      // tab[ i ] *= 2;
      printf( "Tab[ %d ] = %d \n", i+1 , tab[ i ] );
   }
```

b)
```
int i;                  // dostęp za pomocą adresu i indeksu
for( i = 0; i < ROZMIAR; i++ )
   {
      scanf( "%d", tab + i );                    // &*(tab+i) == tab+i
      *(tab+i) = 2 * *(tab+i);                   // *(tab+i) *= 2;
      printf( "Tab[ %d ] = %d \n", i+1 , *(tab+i) );
   }
```

c)
```
int licznik, *wsk; // dostęp za pomocą wskaźnika i licznika
for(  licznik=0, wsk=tab;  licznik < ROZMIAR;  licznik++, wsk++ )
   {
      scanf( "%d", wsk );
      *wsk = 2**wsk;      // *wsk *= 2;
      printf( "Tab[ %d ] = %d \n", licznik+1 , *wsk );
   }
```

d)
```
int *wsk;               // dostęp za pomocą wskaźnika
for(  wsk=tab;  wsk < tab + ROZMIAR;  wsk++ )
   {            // wsk < &tab[ROZMIAR]   ← adres  "końca tablicy"
      scanf( "%d", wsk );
      *wsk *= 2;
      printf( "Tab[ %d ] = %d \n", wsk–tab+1 , *wsk );
   }
```