- **deklaracja** - informacja dla kompilatora, że dana nazwa reprezentuje obiekt jakiegoś typu, ale nie rezerwuje dla niego miejsca w pamięci.
- **definicja** - to deklaracja przy czym dodatkowo rezerwowane jest miejsce w pamięci. Definicja to miejsce, w którym powołuje się obiekt do życia.
- **inicjalizacja** - nadanie obiektowi wartości w momencie jego narodzin

```
01. int a;          // definicja + deklaracja
02. extern int a;   // deklaracja
03. int a = 0;      // definicja + deklaracja + inicjalizacja
```

# WSKAŹNIKI

**Tak naprawdę wskaźniki zostały wprowadzone
w celu usprawnienia programowania
oraz przyspieszenia działania programu.**

***Zmienna to wydzielony obszar pamięci posiadający własną nazwę oraz określoną wartość***

***Wskaźnik zawiera informację o lokalizacji konkretnego obiektu***

# Czym jest ten wskaźnik?

***Wskaźnik to zmienna (zwana zmienną wskaźnikową), która zawiera adres pierwszej komórki pamięci, w której przechowywana jest inna zmienna. Jeśli zmienna zajmuje więcej niż jedną komórkę pamięci, to wskaźnik wskazuje na pierwszą z tych komórek.***

**Dla każdego wskaźnika określany jest jego typ. Wskaźnik typu *int* wskazuje na zmienną typu *int*. Dzięki temu, kompilator wie ile komórek w pamięci zajmuje zmienna rozpoczynająca się w komórce, na którą wskazuje wskaźnik.**

Ze wskaźnikami i adresami związane są dwa operatory:

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

# *Deklaracja wskaźnika*

**typ *nazwa;**

//gwiazdka przed nazwą zmiennej informuje kompilator o tym, że jest ona wskaźnikiem.
W ten sposób zadeklarowaliśmy wskaźnik „nazwa”. Wskaźnik będzie mógł wskazywać na zmienne typu, dla którego został zdefiniowany.

# przykład

**int zmienna;** *//def zmiennej typu int*

**int *wskaznik;** //deklaracja wskaźnika

**wskaznik = &zmienna;** //Od tej chwili wskaźnik będzie wskazywał na adres, pod którym znajduje się zmienna

# przykład

```
 //deklaracja i inicjacja zmiennej całkowitej i
int i=7;

 // deklaracja i inicjacja zmiennej wskaźnikowej do typu
//całkowitego
int *iwsk=&i;

cout<<„ Wartość jaka przechowuje zmienna i:" <<i
<<" Adres zmiennej i: "  <<&i<<"\n";
cout<<„ Wartość zmiennej iwsk to:" <<iwsk <<endl;
```

**W rzeczywistości w programie rzadko kiedy interesuje nas adres zmiennej. Znacznie częściej interesuje nas wartość zmiennej pod tym adresem.**

**Aby wypisać wartość zmiennej, na którą wskazuje wskaźnik, musimy użyć operatora *.**

## różne znaczenie operatora „ *” w programie

```
int liczba=6; // zmienna typu int
int *wskaznik; // wskaznik do typu int        //(1)
wskaznik=&liczba; // powiazanie wskaznika ze zmienną
                                liczba
cout <<"Wartosc zmiennej liczba wynosi "<<liczba<<'\n';
cout <<"Adres zmiennej wynosi "<< &liczba <<'\n';
cout <<"Wartosc wskaznika wynosi "<<wskaznik<<'\n';
cout <<"wartosc uzyskana ze wskaznika wynosi "<<
*wskaznik <<'\n';              // (2)
```

# DO ZAPAMIĘTANIA

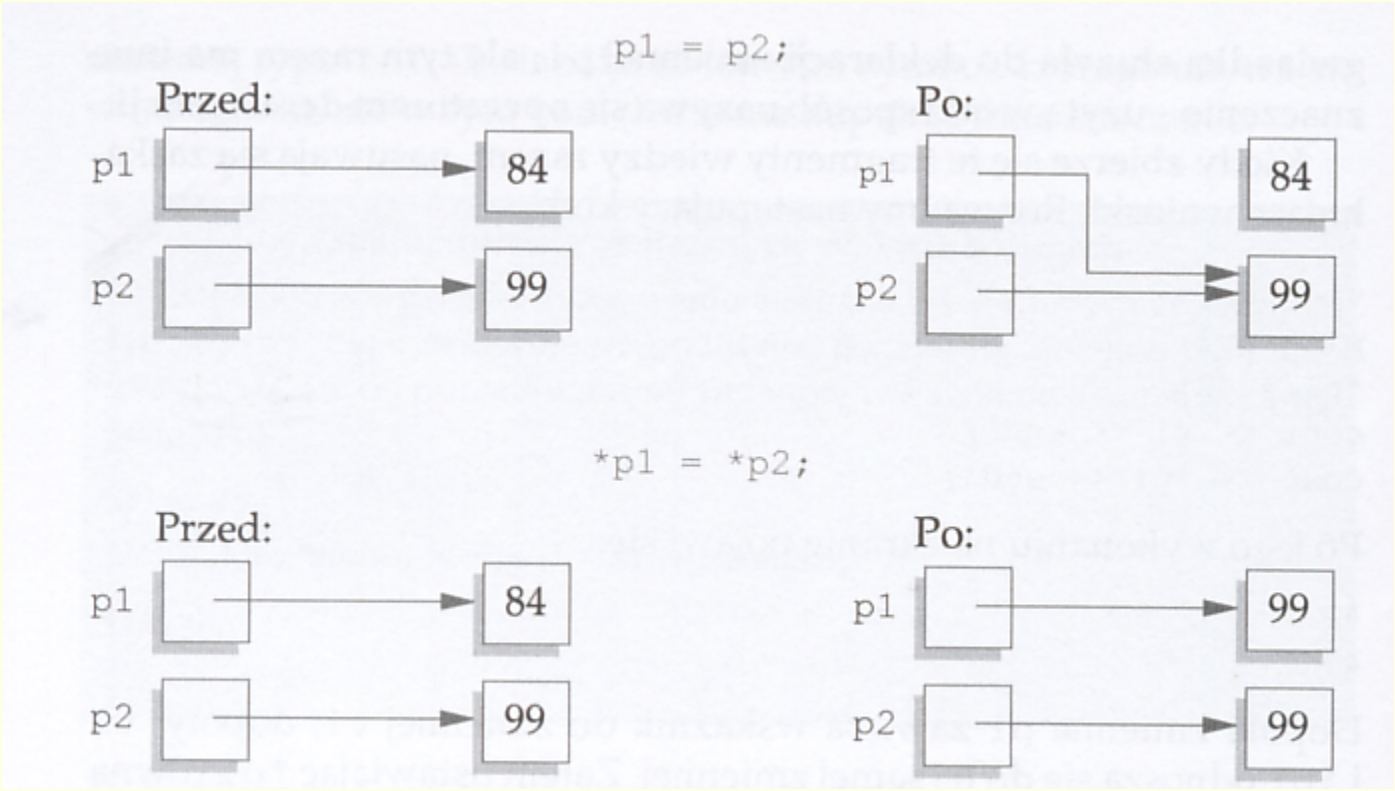
*wsk oznacza wartość, która znajduje się pod adresem będącym wartością wskaźnika wsk

wsk oznacza adres, czyli wartość, wskaźnika o nazwie wsk

&a oznacza adres komórki, w której znajduje się zmienna a

# Użycie operatora przypisania

*(program dwa_wskazniki)*



1. Przeanalizuj poniższy fragment kodu i podaj, jakie wartości zostaną wyświetlone na ekranie monitora:

```
int a, b;
int *wsk1, *wsk2;
wsk2 = &a;

a = 5;
b = 6;
wsk1 = wsk2;
b = *wsk1+3;
wsk1 = wsk2+2;
cout << wsk1;
cout << wsk2;
```

# ZADANIE

1.Napisz program definiujący zmienną typu int oraz wskaźnik do zmiennej typu int. Program powinien wczytać z klawiatury wartość i podstawić ją do zmiennej stosując wskaźnik i operator adresu

2. Napisz program przedstawiający użycie operatora podstawiania

# DO ZAPAMIĘTANIA

**Instrukcja *wsk++; modyfikuje wartość zmiennej, na którą wskazuje wskaźnik**

**Instrukcja wsk++; modyfikuje wartość wskaźnika, czyli adresu, zwiększając tę wartość o tyle bajtów, ile zajmuje zmienna, na którą wskaźnik może wskazywać.**

```
typ NazwaFunkcji( typ NPar1, typ NPar2, )
{
  // ciało funkcji
}
```

- Przekazywanie parametrów przez wartość:

```
typ NazwaFunkcji( typ NazwaParametru ){}
```

- Przekazywanie parametrów przez referencje:

```
typ NazwaFunkcji( typ &NazwaParametru ){}
```

- Przekazywanie parametrów przez wskaźnik:

```
typ NazwaFunkcji( typ *NazwaParametru ){}
```

# Przekazywanie argumentów funkcji przez wskaźnik

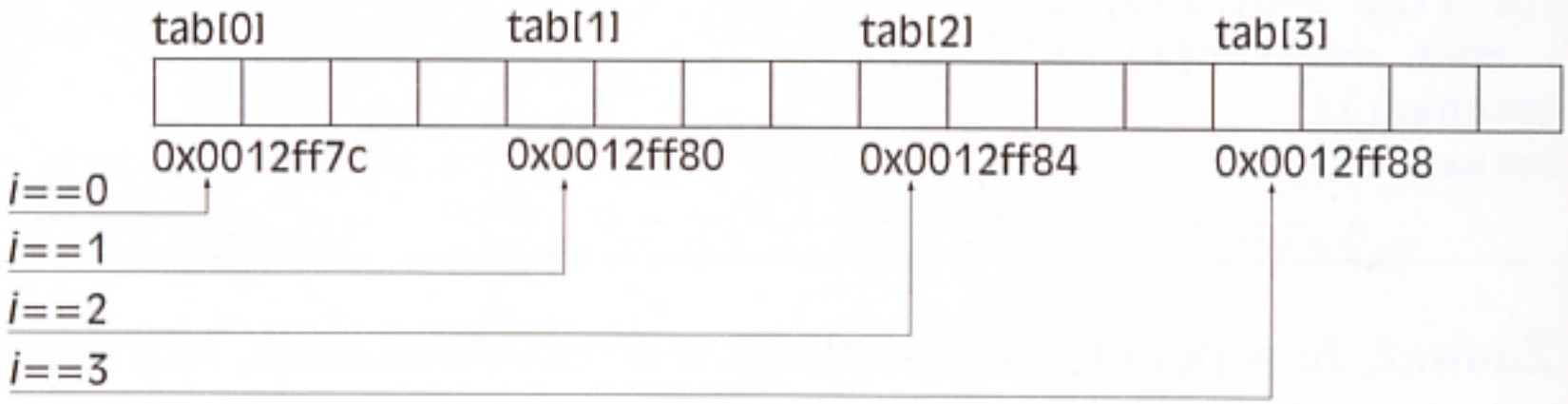
```
#include <bits/stdc++.h>

using namespace std;

void zmien(int *wsk)          // definicja funkcji
{
  *wsk = *wsk+10;
}
void przezadres(int &d)      // definicja funkcji
{
  d = d+10;
}

int main()
{
  int a = 3;
  int *wsk;
  wsk = &a;
 int b = 5;
  int *wskb;
  wskb = &b;
  cout << "Zmienna 'a' ma wartosc: " << a << endl;
  zmien(wsk);                                  // wywołanie funkcji
  cout << "Teraz zmienna 'a' ma wartosc: " << a<<endl;
  cout << "Zmienna 'b' ma wartosc: " << b << endl;
  zmien(wskb);                                                  // wywołanie funkcji
  cout << "Teraz zmienna 'b' ma wartosc: " << b;
  cout << "Teraz przez adres  ";
  przezadres(b);
  cout << endl<<"Teraz zmienna 'b' ma wartosc: " << b;

  return 0;
}
```

# *Zastosowanie wskaźników w tablicach*

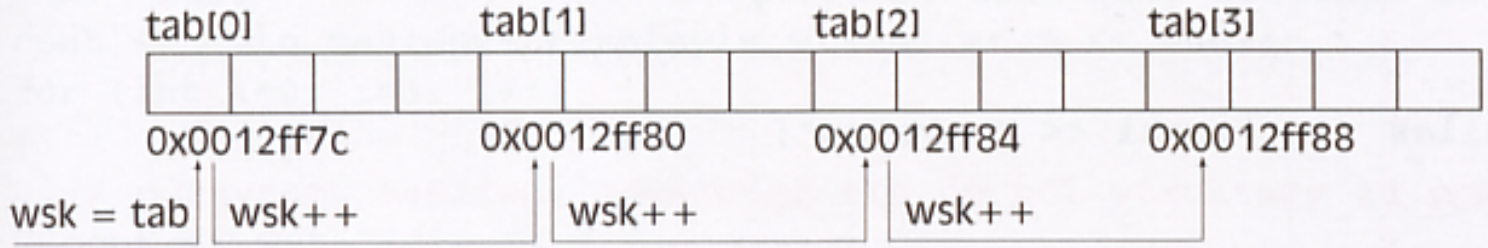
**Nazwa tablicy jest jednocześnie wskaźnikiem do jej pierwszego elementu.**

**tab jest tym samym, co &tab[0]**

**Tablice nie są przekazywane do funkcji z wykorzystaniem operatora pobierania adresu ponieważ sama nazwa tablicy określa położenie w pamięci komputera.**




**Ryc. 8.1.** Fragment obszaru pamięci z tablicą `tab`, w której każdy element zajmuje 4 komórki pamięci, wraz z ilustracją odwoływania się do komórek tablicy przez indeks *i*.




**Ryc. 8.2.** Fragment obszaru pamięci z tablicą `tab`, w której każdy element zajmuje 4 komórki pamięci, wraz z ilustracją inkrementacji wskaźnika `wsk`.

*Program „tabl_przez_wsk”*

2. Wypełnij tablicę dziesięcioelementową losowymi wartościami z przedziału ⟨5, 11⟩. Z wykorzystaniem zdefiniowanego wskaźnika, znając rozmiar tablicy, odczytaj jej elementy w odwrotnej kolejności, aniżeli zostały zapisane.
3. Dwie dwudziestoelementowe tablice wypełnij wartościami losowymi z przedziału ⟨0, 7⟩. Za pomocą dwóch zdefiniowanych wskaźników wyświetl elementy obydwu tablic oraz zbadaj, ile jest par elementów, które znajdują się na tej samej pozycji w tablicach i mają tę samą wartość.

# Wskaźniki i tablice znaków

***przykład***

Program pobiera z klawiatury ciąg znaków. Na ekran monitora zostaje wyprowadzona informacja o liczbie pobranych znaków (wraz z białymi znakami).

# *Wskaźniki i zmienne strukturalne*

**Do poszczególnych pól struktury odwołujemy się za pomocą operatora kropki.**

**Jeżeli do pól struktury odwołujemy się z wykorzystaniem wskaźnika, zastosujemy operator ->**

**przykład**

**W programie utworzymy listę trzech uczniów, a następnie wyświetlimy ją na ekranie monitora**

**Wskaźnik** to zmienna (zwana **zmienną wskaźnikową**), która zawiera adres pierwszej komórki pamięci, w której przechowywana jest inna zmienna. Jeśli zmienna zajmuje więcej niż jedną komórkę pamięci to wskaźnik wskazuje na pierwszą z tych komórek.

Dla każdego **wskaźnika** określany jest jego **typ**.

***Wskaźnik typu int wskazuje na zmienną typu int***.

Dzięki temu, kompilator wie ile komórek w pamięci zajmuje zmienna rozpoczynająca się w komórce, na którą wskazuje wskaźnik.

# ZMIENNE I STRUKTURY DYNAMICZNE

Zmienne dynamiczne są to zmienne, które tworzymy w trakcie działania programu za pomocą operatora **new**
Usuwa się je operatorem **delete**.
Zmienne dynamiczna nie ma własnej nazwy, dlatego dostęp do niej jest możliwy wyłącznie przez adres obszaru pamięci, który ona zajmuje.
Adres ten jest przechowywany we wskaźniku do tej zmiennej. Jeżeli zadeklarujemy wskaźnik do zmiennej dynamicznej, to najlepiej jest nadać mu wartość początkową NULL, czyli przypisać mu adres zerowy.
W ten sposób można się zabezpieczyć przed wykonaniem operacji na obszarze pamięci, którego zmiana wartości nie była zamierzona.

# DYNAMICZNA ALOKACJA PAMIĘCI - TWORZENIE

- tworzenie zmiennej:

```
int *zm;
zm = new int;
```

**New – jest operatorem służącym do alokacji (czyli rezerwowania) obszaru pamięci**

- tworzenie tablicy jednowymiarowej:

```
float *tab;
tab = new float[rozmiar];
```

- tworzenie tablicy dwuwymiarowej:

```
float **tab = new float *[LiczbaWierszy];
for( i=0; i<LiczbaWierszy; i++)
  tab[i] = new float[LiczbaKolumn];
```

## Tablica dynamiczna-

**to taka tablica, która zostaje utworzona w trakcie działania programu i może być usunięta przed jego zakończeniem.**

***Przykład deklaracji:***

```
float *tab=NULL//na razie wskaźnik nic nie
                wskazuje
tab= new float[k]
```

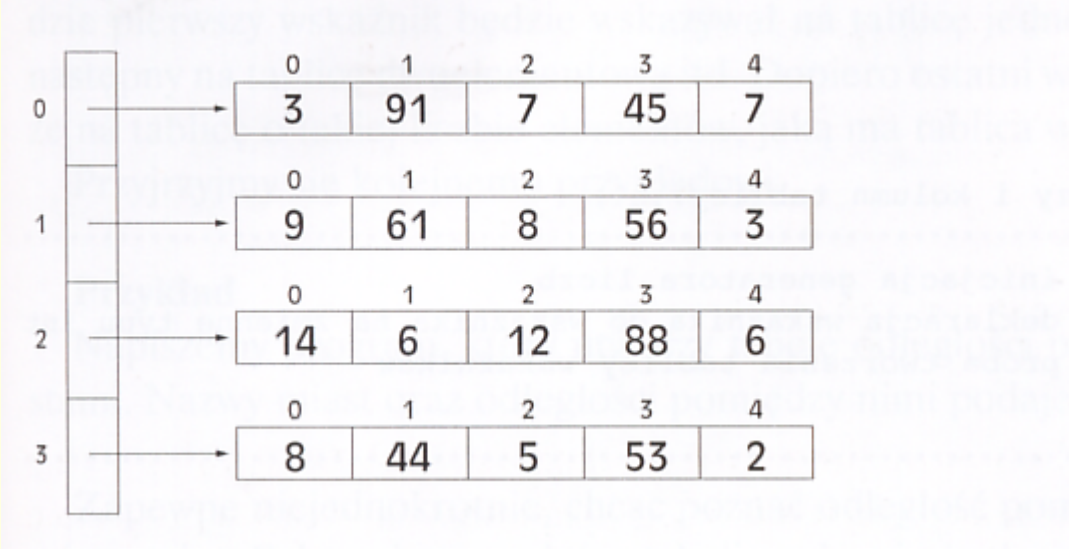
***lub w jednej instrukcji:***

```
float *tab= new float [k]//k może być
                liczbą wpisaną z klawiatury
```

Z tablicy dynamicznej korzystamy jak z tablicy statycznej. Choć tablica jako zmienna dynamiczna nie ma własnej nazwy, to wskaźnikiem, który posłużył do jej utworzenia możemy się posługiwać jak nazwą tablicy statycznej. Poprawna jest np. instrukcja tab[3]=12;

# Tablica dynamiczna dwuwymiarowa

**Zagadnienie tablicy dwuwymiarowej wymaga umiejętności zadeklarowania wskaźnika do wskaźnika.**
**Dwuwymiarowa tablica dynamiczna będzie jednowymiarową tablicą dynamiczną, przechowującą wskaźniki do tablic dynamicznych jednowymiarowych, w których zostaną umieszczone dane.**



Deklaracja dwuwymiarowej tablicy dynamicznej [w][k] liczb całkowitych wygląda natępująco:

Deklaracja dwuwymiarowej tablicy dynamicznej

```
int **tab;              // tab jest wskaźnikiem do wskaźnika na zmienne
                        // typu int

tab = new int *[w];     // w – liczba wierszy (z klawiatury), tu jest
                        // tablica jednowymiarowa wskaźników do zmiennych
                        // typu int

for (i=0; i<w; i++)     // dla każdego wskaźnika z poprzedniej
  tab[i] = new int [k]; // tablicy tworzymy tablicę liczb całkowitych
                        // k – liczba kolumn
```

*jako pierwsza tworzona jest jednowymiarowa tablica wskaźników, dopiero potem powstają tablice, na które te wskaźniki będą wskazywały.*

*Usuwanie z pamięci będzie się odbywać w odwrotnej kolejności. Jako pierwszą zwolnimy pamięć zajmowaną przez tablice przechowujące liczby, następnie tablice wskaźników, które na te tablice wskazywały.*

# Zadanie

**Napisz program, który utworzy dwuwymiarową tablicę dynamiczna o żądanych wymiarach (podanych z klawiatury) i wypełni ją losowymi liczbami ze zbioru {0,1,2,…,100}**

1. Napisz program, który utworzy $n$-elementową tablicę dynamiczną, gdzie $n$ jest wartością podaną przez użytkownika, wypełni ją liczbami rzeczywistymi, również podanymi przez użytkownika, a następnie skopiuje do nowo utworzonej tablicy dynamicznej tylko te liczby, które są większe od 0.
2. Napisz program, który utworzy dwie kwadratowe tablice dynamiczne o rozmiarze podanym przez użytkownika, wypełni je losowymi liczbami z przedziału $\langle 1, 9 \rangle$, a następnie wyświetli informację o liczbie elementów, które stoją na tych samych pozycjach w obu tablicach i mają te same wartości.

## I sposób

```
int **tab;  // deklaracja wskaźnika do wskaźnika na zmienne typu int

 tab = new int *[wie];

  tab[i] = new int [kol];


 for (i=0; i<wie; i++)
   {for (j=0; j<kol; j++)

     {tab[i][j] = rand()%10;        // wypełnianie tablic liczbami
     cout << setw(4) << tab[i][j];          }// wyświetlanie
elementów
    cout << endl;
    }
```

## II sposób

```
int **tab;  // deklaracja wskaźnika do wskaźnika na zmienne typu int

    tab = new int *[wie];

     tab[i] = new int [kol];


 for (i=0; i<wie; i++)
    {for (j=0; j<kol; j++)

      {*(*(tab+i)+j) = rand()%10;   // wypełnianie tablic liczbami
   cout << setw(4) << *(*(tab+i)+j);}// wyświetlanie    elementów
    cout << endl;
    }
```