***Wskaźniki***

**Definicja.** Wskaźnik to zmienna przechowująca adres innej zmiennej.

Inaczej, jest to zmienna wskazująca na adres innej zmiennej. Stąd nazwa wskaźnik. Potocznie mówi się: wskaźnik **wsk** pokazuje na zmienną **x**. Oznacza to, że w zmiennej **wsk** jest zapamiętany adres zmiennej **x**.

**Tworzenie wskaźnika**

Zmienną wskaźnikową tworzymy podobnie jak zwykłe zmienne pamiętając tylko, aby przed nazwą wpisać operator "\*".

int \*a, \*b;

float \*wskf, \*wskf1;

string \*wsknapis;

**Przypisywanie wskaźnikowi adresu zmiennej**

Aby podstawić do wskaźnika adres zmiennej należy użyć operatora "&". Potocznie mówi się: Aby ustawić wskaźnik na zmienną.

int x=123, y=-45;

int \*wskx=&x;

int \*wsky=&y;

double z=2.99;

double wskz=&z;

string napis=”INFORMATYKA”;

string \*wskst=&napis;

**Wypisywanie adresu i wartości zmiennej**

**Wyświetlając zawartość wskaźnika zobaczymy adres komórki pamięci, która jest w nim zapamiętana (czyli adres zmiennej, na którą wskaźnik pokazuje). Aby wyświetlić wartość zmiennej, którą wskaźnik pokazuje należy użyć operatora wyłuskiwania czyli „\*”;**

int x=123;

int \*wskx=&x;

cout<<x<<endl; //”zwykłe” pokazanie zawartości zmiennej x

cout<<wskx<<endl; //pokazanie adresu ze wskaźnika

cout<<\*wskx<<endl; //pokazanie zawartości zmiennej x poprzez wskaźnik

**Ćwiczenie 1 Umieść w programie zmienne: int i=12, float f=3.52 oraz string st=”Programowanie”. Zdefiniuj trzy wskaźniki i ustaw je na odpowiednie zmienne. Wypisz na ekran zawartość zmiennych w „zwykły” sposób oraz poprzez wskaźnik. Zmierz ile bajtów pamięci zajmuje każdy ze wskaźników (wskazówka: int rozmiar = sizeof(wsk)).**

**Wniosek: (zapisz) …………….**

**Zmienne dynamiczne – zastosowanie wskaźników**

**W momencie uruchomienia programu system operacyjny przydziela pamięć (proces przydzielania pamięci nazywać będziemy alokacją)** **:**

* **na kod programu (czyli listę rozkazów dla procesora)**
* **na zmienne statyczne, czyli te definiowane w sposób już Wam znany, np.**

int i;   // potrzebne 4 bajty - tyle zajmuje liczba całkowita

char zn; // potrzebny 1 bajt - znak w kodzie ASCII

int t[10][10]; //potrzebne 10x10x4 = 400 bajtów

Jeżeli pamięci na te zmienne nam zabraknie - program po prostu się nie uruchomi. Pamięć przydzielona na ***zmienne statyczne*** na początku działania programu pozostaje przez nie zajęta aż do końca jego pracy - i nic nie możemy z tym zrobić. Nakłada to na programistę poważne ograniczenia - już w momencie pisania programu musimy przewidzieć maksymalne zapotrzebowanie na pamięć.

Aby uniknąć wszelkich ograniczeń wymienionych powyżej, wymyślono **zmienne dynamiczne**. Zamiast na początku, podczas uruchamiania programu, alokować pamięć "z zapasem", można poprosić system o przydzielenie nam pamięci na taką zmienną dokładnie w tym momencie, w którym nam to będzie potrzebne. Kiedy skończymy z niej korzystać, zawiadomimy o tym system i pamięć do tej pory przez nas zajęta zostanie mu zwrócona do ponownego wykorzystania, bądź to przez nas, bądź przez inny program.

**Zmienne dynamiczne są to zmienne tworzone operatorem *new* podczas wykonywania programu i istniejące aż do ich jawnego skasowania operatorem *delete*.**

Kasowanie zmiennych dynamicznych jest bardzo istotne - jeśli tego nie zrobimy, pamięć na te zmienne pozostanie zarezerwowana nawet po zamknięciu programu. W ten sposób będzie następował tzw. ***wyciek pamięci***; przy każdym kolejnym uruchomieniu programu będą zajmowane następne obszary pamięci i w końcu szybko może jej zabraknąć.

**Zmienne statyczne są tworzone w momencie uruchamiania programu i istnieją do końca jego pracy. Zmienne dynamiczne tworzone i usuwane są w trakcie pracy programu poprzez wywoływanie odpowiednich operatorów (new i delete). Do zmiennych statycznych zwykle odwołujemy się przez ich nazwę, do zmiennych dynamicznych zwykle odwołujemy się przez wskaźnik.**

Operator **new *nazwa\_typu*** przydziela pamięć na zmienną dynamiczną takiego typu, jaki wskazuje *nazwa\_typu*. Adres przydzielonej pamięci jest zwracany jako wynik. Jeżeli alokacja (przydział pamięci) się nie powiedzie, np. pamięci już nie wystarczy, zostanie wstawiony adres zerowy, oznaczany w C++ poprzez ***NULL***.

Operator **delete *zmienna\_wskaźnikowa*** odbiera pamięć zarezerwowaną na zmienną dynamiczną umieszczoną pod adresem pamiętanym w *zmienna\_wskaznikowa*.

**Przykład**

int \*wsk\_i;

wsk\_i = new int;

\*wsk\_i = 20;   //umieszczenie wartości w zm.dynamicznej

cout << "Wartosc wskazywana =" << \*wski<< endl;  //wypisanie zm.dynamicznej

delete wsk\_i;

**Ćwiczenie 2 Rozbuduj powyższy przykład o inne zmienne dynamiczne typu int, long, char, string, itp.**

**Tablice dynamiczne – zastosowanie wskaźników**

Wszystkie tablice, których używaliśmy dotychczas, są tablicami statycznymi, gdyż pamięć na nie jest rezerwowana na etapie kompilacji, zanim program zacznie się wykonywać. Dlatego właśnie rozmiar ich jest określany za pomocą stałych, w dobrze znany Wam sposób:

int main()

{

const int n=200;

int T [n] ;

}

Jeśli jednak nie wiemy z góry, ile elementów będziemy umieszczać w tablicy, musimy ją rezerwować niepotrzebnie dużą, na zapas (podając dużą wartość stałej n). Aby temu zaradzić, aż się prosi, by najpierw dowiedzieć się od użytkownika, ile elementów ma mieć tablica, a potem dopiero przydzielić jej pamięć. To oznacza jednak, że taką rezerwację należy przeprowadzić w trakcie działania programu - będzie to więc **dynamiczna rezerwacja pamięci**.

int main( )

{

int n;

cout << "ile elementow ma być w tablicy ? " << endl;

cin >> n;

int \*tab = new int [n];

// rezerwujemy pamięć  dla n elementów tablicy typu int

// i adres pierwszego z tych elementów  zapamiętujemy jako tab

for (int i=0; i<n; i++) cin >> tab[i]; // wykonujemy działania na tablicy

delete [] tab; // na końcu  zwalniamy pamięć  przydzieloną na tablicę

}

**Ćwiczenie 3 Zapisz algorytm sortowania bąbelkowego dla tablicy dynamicznej.**

**Tablice dynamiczne – zapis indeksowy i wskaźnikowy**

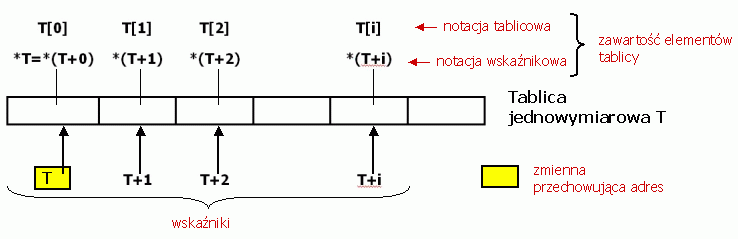
Dla tablic w C++ obowiązuje zasada:

**Nazwa tablicy jest jednocześnie adresem jej pierwszego elementu.**

Przykład

int tab[50]; //tablica statyczna

int \*wsktab=tab;//to samo ale dłużej: int \*wsktab=&tab[0];



W notacji wskaźnikowej wystarczy inkrementować wskaźnik aby przesuwać się do kolejnego elementu tablicy.

int tab[50]; //tablica statyczna

int \*wsktab=tab;//to samo ale dłużej: int \*wsktab=&tab[0];

wsktab++;

cout<<\*wsktab; //to samo co cout<<tab[1];

**Ćwiczenie 4 Zapisz dla porównania w zapisie indeksowym i wskaźnikowym wczytywanie danych z klawiatury do tablicy 10-cio elementowej oraz wypisanie tej tablicy na ekran.**

Opracowano na podstawie: <http://iair.mchtr.pw.edu.pl/~bputz/aisd_cpp/lekcja4/segment1/main.htm>

Warto zobaczyć: http[://miroslawzelent.pl/kurs-c++/wskazniki-alokowanie-pamieci/](http://miroslawzelent.pl/kurs-c++/wskazniki-alokowanie-pamieci/)