Część XXI C++ w

Wskaźniki

Zmienna wskaźnikowa jest dosyć ciekawym typem danych używanym przez początkujących sporadycznie. Jednak bardziej zaawansowany programista bez wskaźników nie może się obyć.

Jaka jest funkcja wskaźników (po ang. pointers)? Zgodnie z nazwą -wskazują miejsce, w którym znajduje się jakiś obiekt (stała, zmienna, tablica czy funkcja). Innymi słowy, zmienna wskaźnikowa przechowuje adres komórki pamięci naszego komputera, w której może znajdować się na przykład wartość innej zmiennej.

Oto kilka najważniejszych zastosowań wskaźników:

► znajomość adresu danych w pamięci komputera i dzięki temu odwołanie się do nich za pomocą tego adresu, to w wielu wypadkach najszybszy sposób na dotarcie do tych danych,

► możemy wywoływanym funkcjom przekazywać adres w pamięci, pod którym przechowywane są wartości zmiennych czy innych (czasem bardzo dużych) struktur danych. Dzięki temu funkcje mogą operować na tych zmiennych (a nie tylko ich wartościach), a w wypadku przekazania wskaźnika do dużych struktur (na przykład tablic), program zajmuje mniej pamięci i wykonuje się znacznie szybciej.

Adres a wartość zmiennej

Zanim używać wskaźników, przypomnijmy sobie zasadę działania zmiennej. Dzięki temu łatwiej będzie nam zrozumieć istotę wskaźników.

Ćwiczenie 1

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko74**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int x = 11;

    cout << "Wartosc prawostronna: " << x << endl;
    cout << "Wartosc lewostronna: " << &x << endl;


    cout << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Po zdefiniowaniu (zadeklarowaniu i zainicjowaniu) zmienna **x** ma dwie wartości: lewostronną i prawostronną. Prawostronną wartością (zwaną po prostu wartością) jest liczba 11. Wartość lewostronna to adres komórki w pamięci, w której przechowywana jest wartość prawostronna. Tak więc nazwa zmiennej (w tym wypadku **x** jest wskazaniem miejsca w pamięci, w której należy szukać wartości zmiennej. Miejsce w pamięci przydzielane jest na stałe w momencie zadeklarowania zmiennej.

Wartość prawostronną (dla uproszczenia będziemy posługiwać się samym zwrotem wartość zmiennej) łatwo sprawdzić - wystarczy wyświetlić ją na ekranie za pomocą na przykład polecenia.

Jak sprawdzić wartość lewostronną (dla uproszczenia będziemy posługiwać się określeniem adres zmiennej)? Również możemy go wyświetlić, wykorzystując do tego operator adresowy & .

3. Po skompilowaniu i uruchomieniu naszego kodu na ekranie zobaczymy

```
Wartosc prawostronna: 11
Wartosc lewostronna: 0x22ff74
```

Wiemy już, jak uzyskać adres zmiennej, Potrafimy go nawet wyświetlić na ekranie. Dlatego zobaczmy, co możemy z uzyskanym adresem zrobić pożytecznego #r-' _

Ćwiczenie nr 2

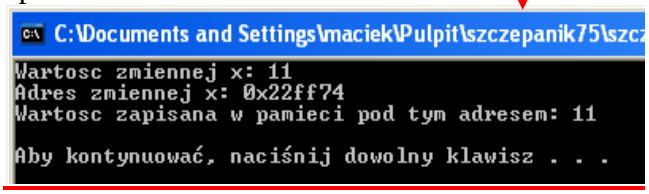
1.Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko75**
2.Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int x = 11;

    cout << "Wartosc zmiennej x: " << x << endl;
    cout << "Adres zmiennej x: " << &x << endl;
    cout << "Wartosc zapisana w pamieci pod tym adresem: " << * &x;


    cout << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Kod od poprzedniego różni się jedynie linijką.

Wyświetlamy w niej wartość zapisaną pod adresem uzyskanym za pomocą operatora &. Używamy do tego operatora *.

3.Z menu Uruchom wybieramy Kompiluj i uruchom
Po skompilowaniu i uruchomieniu kodu, operator ten spowoduje, że zamiast adresu pamięci zostanie wyświetlona wartość pod nim zapisana. Działanie operatora * jest więc dokładnie przeciwne do działania operatora &.



Zmienna wskaźnikowa

Przedstawiony w poprzednich dwóch kodach zapis **x** to nic innego, jak przykładowa wartość zmiennej wskaźnikowej (zmiennej typu wskaźnikowego). Czas dowiedzieć się, w jaki sposób zadeklarować zmienną, aby taką wartość można było jej przypisać

Ćwiczenie nr 3

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko76**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

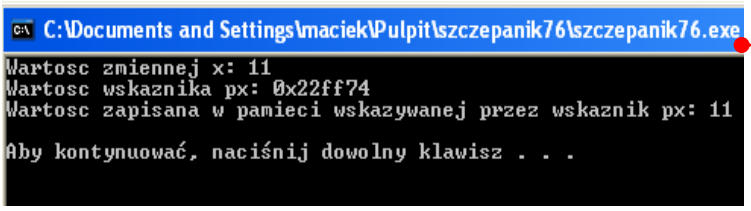
Oprócz zmiennej całkowitoliczbowej **x** deklarujemy również zmienną wskaźnikową **px**, która będzie wskazywała na typ całkowitoliczbowy. Zwróćmy uwagę na operator *, który poprzedza nazwę zmiennej wskaźnikowej - to właśnie ten symbol informuje kompilator, że zmienna (w tym wypadku **px**) jest wskaźnikiem.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int x = 11;
    int *px;
    px = &x;

    cout << "Wartosc zmiennej x: " << x << endl;
    cout << "Wartosc wskaznika px: " << px << endl;
    cout << "Wartosc zapisana w pamieci wskazywanej przez wskaznik px: " << *px;


    cout << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Skoro mamy już zmienną wskaźnikową, możemy jej przypisać adres zmiennej **x**. Z poprzednich przykładów wiemy już, jak uzyskać taki adres. Wystarczy więc użyć operatora przypisania (symbolizowanego przez znak =).

Tak jak poprzednio wyświetlamy wartość zmiennej **x** oraz wartość zmiennej wskaźnikowej **px** (będzie to adres zmiennej **x**). Na końcu wyświetlamy również wartość wskazywaną przez adres będący wartością zmiennej wskaźnikowej **px**. Mówiąc prościej, wyświetlamy daną zapisaną w komórce pamięci o adresie przechowywanym w zmiennej wskaźnikowej **px**

3.Z menu Uruchom wybieramy Kompiluj i uruchom



Po kompilacji i uruchomieniu programu na ekranie zobaczymy widok.
Widzimy więc, że wskaźnik przechowuje adres pamięci, a odwołanie się do informacji zapisanej pod tym adresem uzyskamy za pomocą wyłuskania (czyli wykorzystania operatora *).

Operację, którą wykonuje operator &, można określić mianem pobierania adresu. Operację przeciwną, czyli taką, którą wykonuje operator *, nazywamy dereferencją albo prościej wyłuskaniem (wyłuskujemy pewną wartość z obszaru pamięci). Warto przyswoić sobie te nazwy

W różnych kodach możemy spotkać się z różnym formatem zapisu deklaracji zmiennej wskaźnikowej
Wszystkie są poprawne, jednak proponuje trzymać się jednej formy zapisu - najlepiej

```
int *px;

int* px;

int * px;
```

Zmiana wartości za pomocą wskaźnika

Korzystając ze wskaźników, możemy nie tylko odczytywać dane znajdujące się pod wskazywanym przez nie adresem pamięci, ale także modyfikować te dane. W jaki sposób to robić? Dokładnie tak samo, jak w wypadku zwykłych zmiennych.

Ćwiczenie 4

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko77**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    float x, *px;

    x = 7.456;
    px = &x;

    cout << "Wartosc x: " << x << endl;
    cout << "Wartosc px: " << px << endl;
    cout << "Wartosc wskazywana przez px: " << *px << endl;

    *px = 2.34 + 1;

    cout << "Wartosc x: " << x << endl;
    cout << "Wartosc px: " << px << endl;
    cout << "Wartosc wskazywana przez px: " << *px << endl;

    cout << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

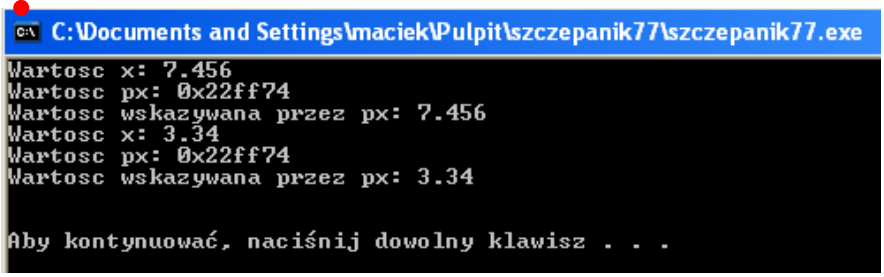
Deklarujemy zmienną **x** typu float oraz wskaźnik **px** tego samego typu

Zmiennej **x** przypisujemy dowolną wartość, a wskaźnikowi **px** adres zmiennej **x**

Wyświetlamy na ekranie wartość zmiennej **x**, adres zapisany w zmiennej wskaźnikowej **px** oraz wartość znajdującą się pod tym adresem w pamięci komputera.

W linii zmieniamy wartość komórki pamięci znajdującej się pod adresem przechowywanym we wskaźniku **px.**
Oczywiście wykorzystujemy do tego operator wyłuskania oznaczany symbolem gwiazdki *. Gdybyśmy go nie wpisali, nie zmienilibyśmy wartości zapisanej pod adresem, tylko zmodyfikowalibyśmy sam adres, który zapisany jest we wskaźniku o nazwie **px**

Aby sprawdzić, co się stało po wykonaniu linii, ponownie wyświetlamy wartości zmiennych **x** i **px** oraz zawartość komórki pamięci wskazywanej przez wskaźnik **px**. **Po kompilacji i uruchomieniu programu** na ekranie widzimy. Możemy się więc przekonać, że zmiana zawartości komórki pamięci powoduje również zmianę wartości zmiennej związanej z tą komórką.



Wszystkie pliki z nazwiskiem i kolejnym numerem umieszczamy w swoim folderze nazwiskocplusplus na serwerze.