Część XXII C++ w

Wskaźniki a tablice

Wskaźniki i tablice są ze sobą w języku C++ ściśle związane. Aby się o tym przekonać wykonajmy cwiczenie.

Ćwiczenie 1

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko78**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

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

using namespace std;

int main(int argc, char *argv[])
{
    int tablica[5]={2, 20, 200, 2000, 20000};
    int *ptablica1, *ptablica2;

    ptablica1 = &tablica[0];
    ptablica2 = tablica;

    cout << "Wartosc wskaznika ptablica1: " << ptablica1 << endl;
    cout << "Wartosc wskaznika ptablica2: " << ptablica2 << endl;
    cout << "Wartosc dereferencji wskaźnika ptablica2: " << *ptablica2 << endl;
    cout << "Wartosc wskaznika (ptablica2+1): " << ptablica2 + 1 << endl;
    cout << "Wartosc dereferencji wskaźnika (ptablica1+1): " << *(ptablica2 + 1) << endl;
    cout << "Wartosc dereferencji wskaźnika (ptablica1+2): " << *(ptablica2 + 2) << endl;


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

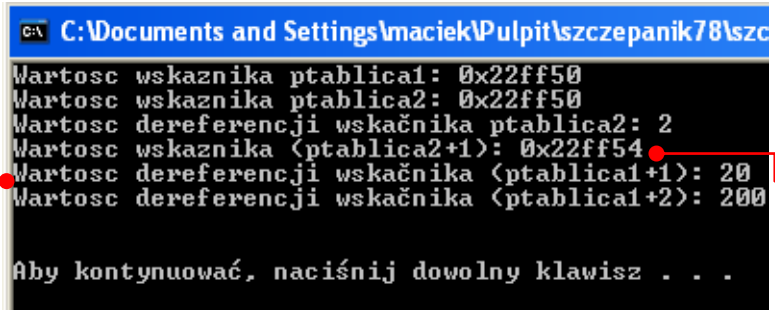
Analiza programu

Definiujemy tablicę pięcioelementową, przechowującą wartości całkowitoliczbowe. Deklarujemy również dwa wskaźniki **ptablica1** i **ptablica2** (oczywiście tego samego typu co tablica)

Do wskaźnika **ptablical**_ przypisujemy adres pierwszego elementu tablicy (wykorzystując operator adresu &).

W następnej lini do wskaźnika **ptabiica2** przypisujemy zmienną tablicową **tablica**, nie wykorzystując operatora &. Wyświetlamy oba wskaźniki - po skompilowaniu i uruchomieniu programu widać, że oba zawierają identyczny adres

Sprawdźmy wartość zapisaną w pamięci pod adresem przechowywanym we wskaźniku do tablicy. Na ekranie wyświetlona zostaje wartość pierwszego elementu tablicy.



Wiemy już, że wskaźnik do tablicy wskazuje na pierwszy element tablicy.W jaki sposób dostać się do kolejnych elementów zmiennej tablicowej? Spójrzmy na następną linię - wyświetlamy w niej wskaźnik ptablica2 powiększony o 1. Wyświetlony na ekranie adres nie będzie jednak powiększony o 1, ale o 4 bajty.

Dlaczego tak się stało? Ponieważ wskaźnik tablica 2 jest typu int, który na przechowanie w pamięci wartości potrzebuje 4 bajtów.

Na koniec wyświetlamy dereferencję wskaźnika

powiększonego o 1 i 2. Na ekranie zobaczymy wartość drugiego i trzeciego elementu naszej tablicy.
Wiemy już więc, w jaki sposób za pomocą wskaźnika odwoływać się do wszystkich elementów tablicy.

Zwróćmy uwagę, że operator wykonujący wyłuskanie oznaczany jest identycznym symbolem (gwiazdki *) jak operator arytmetyczny mnożenia. Różnica między tymi operatorami, po której zresztą rozpoznaje je kompilator, jest taka, że operator mnożenia jest dwuargumentowy (z jego lewej i prawej strony musi znajdować się wartość lub zmienna), a operator dereferencji - jednoargumentowy.

Za pomocą wskaźników do tablicy możemy oczywiście bez problemu modyfikować wartości elementów tablicy.

Wskaźnik do łańcucha znaków

Skoro nazwa tablicy jest tak naprawdę wskaźnikiem wskazującym adres pierwszego elementu, w wypadku łańcuchów znakowych można więc, zamiast tablicą, posłużyć się samym wskaźnikiem na tablicę. Kompilator zapisze taki ciąg znaków w pamięci tak, jak tablicę. Sprawdzimy to na przykładzie kodu w cwiczeniu2. Tworzymy w nim wskaźnik, któremu od razu przypisujemy ciąg znaków zapisanych w cudzysłowach. Następnie, wykorzystując nazwę wskaźnika, wyświetlamy ciąg znaków.

Ćwiczenie nr 2

1.Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko79**
2.Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

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

using namespace std;

int main(int argc, char *argv[])
{
    char *pnapis = "Codziennie niskie ceny";
    cout << pnapis << endl;


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

3.Z menu Uruchom wybieramy Kompiluj i uruchom

Takiej uproszczonej formy zapisu łańcucha znaków używajmy tylko wtedy, gdy nie zamierzamy w kodzie naszego programu zmieniać treści tego napisu.

Tak więc stworzony przez nas kod jest równoznaczny stworzeniu najpierw tablicy znakowej i przypisaniu jej ciągu znaków, następnie zadeklarowaniu wskaźnika typu znakowego oraz przypisaniu tablicy do tego wskaźnika

```
char napis[] = "Codziennie niskie ceny";
char *pnapis;
pnapis = napis;
```

Przekazywanie wskaźników do funkcji

Do tej pory argumentami naszych funkcji były zmienne typu liczbowego. Przekazanie do funkcji zmiennej typu wskaźnikowego stwarza całkiem nowe możliwości.

Już wiemy, co to są wskaźniki i jak się nimi posługiwać. Dowiedzmy się teraz, co pożytecznego możemy dzięki nim osiągnąć. w jaki sposób przekazywać wskaźniki jako argumenty do różnych funkcji.

Tablica jako argument funkcji

Często w naszych programach będziemy chcieli do oddzielnej funkcji przekazać kilka lub nawet kilkanaście argumentów. Użycie w takim wypadku zmiennych jest bardzo niewygodne i nieoptymalne (zabiera dużo pamięci i czasu procesora). Najlepszym rozwiązaniem w takim wypadku jest więc przekazanie do funkcji tablicy, w której umieszczamy wszystkie wartości przekazywane do funkcji. Spójrzmy na przykład, którego kod znajdziemy w ćwiczeniu 3.

Ćwiczenie nr 3

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko80**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

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

using namespace std;

float sumuj(float tablica[]);

int main(int argc, char *argv[])
{
    float liczby[4];
    float *pliczby = liczby;(1)

    cout << "Podaj 4 liczby oddzielajac je spacja: " << endl;
    for (int i=0; i<4; i++) cin >> liczby[i];(2)

    cout << "Suma podanych liczb wynosi: " << sumuj(liczby) << endl;(3)
    cout << "Suma podanych liczb wynosi: " << sumuj(pliczby) << endl;(4)
//  cout << "Ostatni element tablicy wynosi: " << liczby[3] << endl;(5)
    cout << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}

float sumuj (float tablica[])(6)
{
      float wynik = 0;
      for (int i=0; i<4; i++) wynik += tablica[i];(7)
//    tablica[3] = 100;(8)
      return wynik;
}
```

3.Z menu Uruchom wybieramy Kompiluj i uruchom

Analiza programu

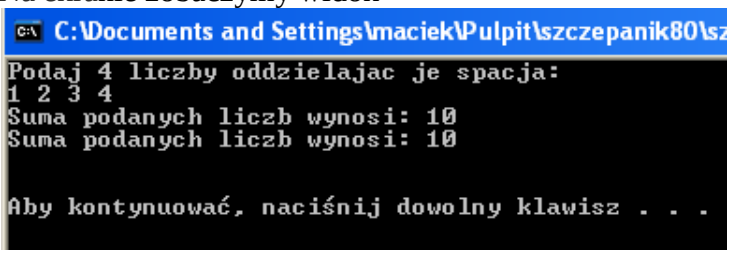
Deklarujemy czteroelementową tablicę **liczby** (przechowującą liczby zmiennoprzecinkowe) oraz wskaźnik o nazwie **pliczby** , któremu przypisujemy tablicę **liczby** 1

Następnie z klawiatury pobieramy cztery liczby 2 i w pętli wpisujemy je do zmiennej tablicowej **liczby**.

W następnych liniach wywołamy funkcje **sumuj ( ) ,** więc najpierw ją napiszmy 6 (oczywiście nie zapominamy o jej deklaracji, którą umieszczamy na samym początku naszego programu). Zadaniem funkcji będzie zsumowanie czterech liczb przekazanych do niej w postaci tablicy i zwrócenie tej sumy. Dodawanie wykonujemy w pętli for 7

Wyświetlamy na ekranie wynik działania funkcji **sumuj ( )**, przekazując do niej nazwę tablicy 3. Wykonajmy jeszcze raz tę samą linijkę, przekazując tym razem do funkcji wskaźnik **pliczby** wskazujący na naszą tablicę 4

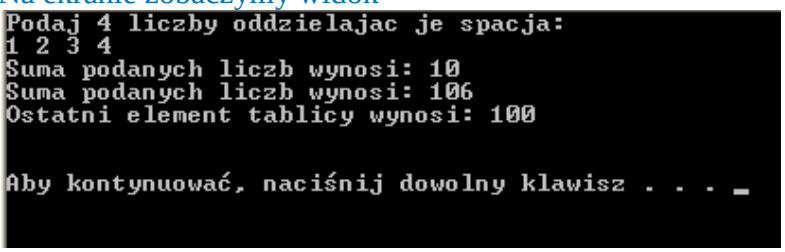
Na ekranie zobaczymy widok



Co z niego wynika? Między innymi to, że nie ma znaczenia, czy do funkcji przekażemy nazwę tablicy, czy wskaźnik do tej tablicy. Obie zmienne wskazują bowiem na ten sam adres w pamięci, w którym znajduje się pierwszy element tablicy

Skoro przekazaliśmy do funkcji adres pierwszego elementu tablicy, to w rzeczywistości podaliśmy funkcji miejsce przechowywania wartości. Przekazanie więc tablicy jako argumentu znacznie się różni od przekazania do funkcji zmiennej - w tym drugim wypadku przekazujemy wyłącznie wartość zmiennej. Sprawdźmy, co się stanie, jeśli wewnątrz funkcji sumuj() zmienimy wartość jednego z elementów naszej tablicy (jak pamiętamy, w wypadku zmiennej zmiana wartości wewnątrz funkcji nie miała wpływu na wartość zmiennej poza tą funkcją). Aby to zrobić, wystarczy usunąć znaki komentarza z linii 5 oraz 8 i uruchomić program.

Na ekranie zobaczymy widok



Widzimy więc, że przekazanie do funkcji tablicy daje możliwość tej funkcji modyfikacji elementów tej tablicy

Wskaźnikowe argumenty

Pamiętamy, że przekazanie do funkcji nazwy zmiennej powoduje tak na prawdę przesłanie do tej funkcji kopii wartości zmiennej. W ten sposób nie da się wykonać na przykład funkcji zamieniającej wartości dwóch zmiennych (wartość zmiennej **a** zostaje przypisana do zmiennej **b** i odwrotnie). Jednak taką funkcję możemy bez problemu napisać, korzystając ze wskaźników. Spójrzmy na poniższe ćwiczenie.

Ćwiczenie nr 4

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko81**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

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

using namespace std;

void zamien1 (int *a, int *b);
void zamien2 (int &a, int &b);

int main(int argc, char *argv[])
{
    int a, b;
    a = 10;
    b = 20;
    cout << "a=" << a << "  b=" << b << endl;
    zamien1(&a, &b);
    cout << "a=" << a << "  b=" << b << endl;
    cout << endl << endl;

    zamien2(a, b);

    cout << "a=" << a << "  b=" << b << endl;

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

void zamien1 (int *x, int *y)
{
     int z = *x;
     *x = *y;
     *y = z;
}

void zamien2 (int &a, int &b)
{
     int x = a;
```

Definiujemy funkcję o nazwie **zamien1()** niezwracającą żadnego wyniku i jako jej argumenty ustalamy dwa wskaźniki.

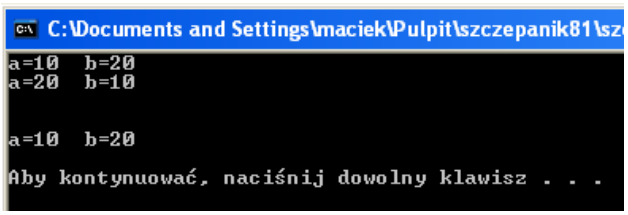
Następnie deklarujemy pomocniczą zmienną **z** i przypisujemy jej wartość komórki wskazywanej przez wskaźnik **x** . Teraz wystarczy już pod adres wskazywany przez wskaźnik **x** przypisać wartość z adresu przechowywanego przez zmienną **y**, do którego z kolei przypisujemy wartość zmiennej **z** (znajduje się w niej stara wartość wskazywana przez wskaźnik **x**).

Aby sprawdzić działanie funkcji **zamien1()** wewnątrz funkcji **main** deklarujemy dwie zmienne i przypisujemy im dowolne wartości

Następnie wyświetlamy wartości zmiennych i wywołujemy stworzoną przed chwilą funkcję **zamien1()**

Ponieważ argumentami funkcji są wskaźniki (a więc zmienne przechowujące adres), nie możemy przekazać nazw zmiennych, tylko odpowiadające im adresy w pamięci (aby uzyskać adres, wykorzystujemy operator &, ,który umieszczamy przed nazwą każdej zmiennej).

Wyświetlając ponownie wartości zmiennych **a** i **b**, przekonamy się łatwo, że funkcja **zamien1()** działa



3.Z menu Uruchom wybieramy Kompiluj i uruchom

Przekazanie przez wskaźnik lub referencję

Niestety, pamiętanie o tym, aby do danej funkcji przekazywać adresy zmiennych, jest niewygodne. Dlatego przekazywanie wartości przez wskaźnik może być kłopotliwe dla niezbyt zaawansowanych programistów. Oczywiście możemy skorzystać z innego rozwiązania - tak zwanego przekazywania

danych przez referencję

```
void zamien2 (int &a, int &b)
{
      int x = a;
      a = b;
      b = x;
}
```

Wtedy wywołanie funkcji będzie wyglądało tak (nie musimy korzystać z operatora &).

```
zamien2(a,b);
```

Nie będziemy tutaj wyjaśniać pojęcia referencji, warto jednak wiedzieć, że coś takiego istnieje.

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