Tablice w C++ - wiadomości wstępne

Na dzisiejszych zajęciach omówimy podstawowe zasady postępowania przy programowaniu z użyciem tablic w języku C++. Na początek przeanalizujmy poniższy listing (plik *tablice.cpp*):

```
#include<iostream>
using namespace std;

void wypisz(float tablica[], unsigned dlugosc);

int main()
{
float dane[5];

   cout<<"Program demonstruje poslugiwanie sie tablicami"<<endl<<endl;
   cout<<"Na poczatku tablica zawiera nastepujace wartosci:"<<endl;
   wypisz(dane, 5);

   cout<<endl;
   cout<<"Teraz cos do niej wpiszemy. Podaj trzy liczby (zatwierdz ENTER'em): "<<endl;
   cin >> dane[0];              // PIERWSZA wartosc w tablicy jest pod indeksem "zero"
   cin >> dane[1];
   cin >> dane[2];

   cout<<endl;
   cout<< "Teraz tablica zawiera nastepujace liczby: "<<endl;
   wypisz(dane, 5);

   cout<< endl<< "Wykonamy teraz pare przypisan... i taki jest efekt: "<<endl;

   dane[3] = dane[0];
   dane[4] = dane[1];
   wypisz(dane, 5);
}

void wypisz(float tablica[], unsigned dlugosc)
{
unsigned indeks;

   for(indeks=0; indeks<dlugosc; indeks++) cout << tablica[indeks] <<endl;
}
```

Powyższy program wyświetli na ekranie (po podaniu przykładowych danych: 123, 234, 345):

```
Program demonstruje poslugiwanie sie tablicami

Na poczatku tablica zawiera nastepujace wartosci:
-1.99977
2.17976
2.39516
2.01139
-1.99977

Teraz cos do niej wpiszemy. Podaj trzy liczby (zatwierdz ENTER'em):

Teraz tablica zawiera nastepujace liczby:
123
234
345
2.01139
-1.99977

Wykonamy teraz pare przypisan... i taki jest efekt:
123
```

234
345
123
234

W prototypie funkcji *wypisz* pojawia się następująca deklaracja argumentu:

float tablica[]

Co ona oznacza? Gdyby nie było pary nawiasów kwadratowych *[]* , sprawa byłaby prosta – *tablica* jest zmienną typu *float*. Nawiasy kwadratowe oznaczają, że dany obiekt jest tablicą. Tablica jest strukturą danych, składającą się z elementów tego samego typu, które identyfikuje się poprzez indeks. Mówiąc obrazowo, tablica dzieli się na komórki, z których każda może pomieścić daną zadeklarowanego typu, zaś każda z komórek jest oznaczona pewną liczbą.

W jaki sposób definiuje się tablice? Do zdefiniowania tablicy potrzebne są trzy informacje: typ elementów tablicy, nazwa tablicy oraz ilość jej elementów. Informacje te są potrzebne kompilatorowi, między innymi po to, aby wyznaczyć ilość pamięci, potrzebnej do przechowywania tablicy. Do uzyskania tej informacji potrzebny jest rozmiar pojedynczego elementu (informację taką można uzyskać za pomocą operatora *sizeof*) oraz ilość elementów tablicy. Reguła definiowania tablicy jest następująca: ilość elementów tablicy podaje się wewnątrz nawiasów kwadratowych, przy czym ilość musi być stałą, znaną na etapie kompilacji. Na przykład definicja

float dane[5];

oznacza, że tablica *dane* zawiera pięć liczb typu *float*. Można tworzyć tablice z elementami dowolnego typu, w szczególności elementem tablicy może być tablica. Używając tablic jako elementów tablic tworzy się tablice wielowymiarowe.

Użytkowanie tablic sprowadza się do odwołania się do pojedynczego elementu tablicy oraz podstawienia tablicy jako argumentu dla funkcji. Odwołania się do elementu tablicy odbywa się poprzez wskazanie jej za pomocą indeksu, będącego liczbą całkowitą. W językach *C* oraz *C++* istnieje kardynalna zasada:

INDEKSOWANIE ELEMENTÓW TABLICY ROZPOCZYNA SIĘ OD ZERA

Powyższe stwierdzenie oznacza, że kolejne elementy *N*-elementowej tablicy są oznaczone indeksami: *0, 1, 2, ..., N - 1* – razem *N* możliwych indeksów. Sprawdzanie, czy indeks elementu jest poprawny, należy do programisty – nie jest sygnalizowane na etapie kompilacji np. odwołanie się do 2000-ego elementu tablicy, która ma 10 elementów. Odwołanie się do elementu tablicy realizowane jest za pomocą wyrażenia:

nazwa_tablicy[indeks_komórki]

Wyrażenie tej postaci może pojawić się wszędzie tam, gdzie może wystąpić zmienna typu, jakiego są elementy tablicy np. zarówno po prawej, jak i po lewej stronie operatora przypisania – oba te przypadki zachodzą w naszym przykładowym programie. Do wyjaśnienia pozostało jeszcze jedno zagadnienie (patrz wydruk zawartości ekranu po uruchomieniu programu przykładowego): skąd wzięły się początkowe wartości elementów tablicy? Zmienne lokalne funkcji, jeśli nie zostaną zainicjalizowane, ani nie zostanie im przypisana wartość, zawierają wartości nieokreślone. Do momentu wykonania pierwszej operacji, która określa przechowywaną w nich wartość, zawierają wartości nieokreślone, choć konkretne. W powyższym programie celowo został popełniony jeden z częstszych błędów: dokonano próby odwołania się do zmiennych (tutaj elementów tablicy), które nie były jawnie zainicjalizowane, przed operacją przypisania im wartości. Proszę zauważyć, że powoduje to niestabilne działanie programu, który jest całkowicie zgodny ze składnią języka i poprawnie się kompiluje! Powoduje to trudności w wyszukiwaniu tego typu błędu. Należy zatem pamiętać, aby przed odczytem zmiennej nadać jej pożądaną wartość: w momencie definiowania (tzw. inicjalizacja zmiennej) lub za pomocą operacji przypisania. Przykłady inicjalizacji tablic znajdziemy w następnym programie (plik *inicjalizacja.cpp*):

```
#include<iostream>
using namespace std;

void wypisz(char napis[]);

int main()
{
char nazwisko[]="Drzewiecki";
float dane[3]= { 184, 72.5, 39 };

   cout<<"Program demonstruje sposoby inicjalizacji tablic."<<endl<<endl;
   cout<<"Dane programisty: "<<endl;
   cout<<"Nazwisko : ";
   wypisz(nazwisko);
   cout<< endl << "Wzrost :" << dane[0] <<endl;
   cout<< "Waga :" << dane[1] <<endl;
   cout<< "Numer kolnierzyka: " << dane[2] <<endl;
}

void wypisz(char napis[])
{
unsigned indeks=0;
char znak;

   while( znak=napis[indeks++] ) cout<<znak;
}
```

Początkowe wartości tablic *nazwisko* oraz *dane* zostały nadane mechanizmem inicjalizacji. Tutaj nie spowodujemy błędu (błędu wykonania programu, nie błędu kompilacji!), odczytując wartości elementów tablic, zanim coś im przypiszemy – nadane są im wartości początkowe, choć oczywiście można je później zmienić, przypisując im nowe wartości. Warta omówienia jest definicja tablicy *nazwisko*: zostało powiedziane, że przy definicji tablicy konieczne jest podanie ilości jej elementów, tutaj zaś nie podano jawnie ilości elementów. W wypadku, gdy nie zostanie podany rozmiar tablicy przy jej definicji, kompilator spróbuje ustalić potrzebny rozmiar tablicy na podstawie wyrażenia inicjalizującego. W przypadku tablicy *nazwisko* będzie to ilość znaków, koniecznych do zapamietania stałej tekstowej, podanej w cudzysłowach.

Jest coś szczególnego, co charakteryzuje *tablice znakowe* w języku *C++* (tak samo jest w języku *C*): oprócz znaków, stanowiących treść użytecznego napisu, w tablicy zaraz za znakami napisu umieszczony jest specjalny znak, tzw. *NULL*, o kodzie równym zero, sygnalizującym koniec napisu. Jest to ważne, gdyż należy uwzględnić ten dodatkowy znak, szacując rozmiar potrzebnej tablicy.

Kolejna definicja tablicy (tablica *dane*) jest przykładem na typową inicjalizację tablic, nie będących tablicami tekstowymi. Proszę zwrócić uwagę na sposób podania wartości inicjalizujących dla elementów tablicy: lista wartości dla kolejnych komórek, rozdzielona przecinkami, jest zamknięta wewnątrz pary nawiasów klamrowych, zaś przed nawiasem otwierającym umieszczony jest znak równości. Lista wartości może zawierać mniej elementów, niż wynosi rozmiar tablicy, ale nie może zawierać więcej wartości niż wynosi rozmiar tablicy. W wypadku podania mniejszej ilości wartości inicjalizujących, niż wynosi rozmiar tablicy, pozostałe elementy tablicy zostaną zainicjalizowane wartościami zerowymi odpowiedniego typu. Inicjalizacja w powyższej formie jest również możliwa do wykonania dla tablic tekstowych (tzw. *string*), dla początkowej zawartości tablicy *nazwisko* byłaby ona postaci:

```
char nazwisko[]={'D', 'r', 'z', 'e', 'w', 'i', 'e', 'c', 'k', 'i', '\0'};
```

, jednak wyraźnie widać, że jest ona w takiej sytuacji mniej wygodna od podania literału tekstowego. Dla porządku przypomnę, że stałe znakowe ujmuje się w parę znaków apostrofu '. Escape sekwencja *\0* oznacza znak o kodzie zero, czyli *NULL* – znak kończący tekst.

Na poprzednich zajęciach zostało omówione m.in. przekazywanie argumentów do funkcji przez referencję. Niejako w związku z tym należy wspomnieć o koronnej zasadzie języków *C* oraz C++, związanej z tablicami:

FUNKCJA, DOSTAJĄC TABLICĘ JAKO SWÓJ ARGUMENT, MA PEŁEN DOSTĘP DO ELEMENTÓW TABLICY – MOŻE Z NICH CZYTAĆ I DO NICH PISAĆ

Nie jest konieczne odwołanie poprzez referencję, aby z wnętrza funkcji zmienić zawartość zewnętrznej komórki tablicy, podanej funkcji jako jej argument. Zasadę tę wprowadzono, aby uniknąć kopiowania (potencjalnie) dużych tablic w trakcie przygotowania kontekstu wywołania funkcji, zawierającej argumenty tablicowe.

W powyższym programie do wypisania zawartości tablicy *nazwisko została* użyta odrębna funkcja. Nie jest to konieczne, wypisywanie tekstu na ekran przez strumień *cout* znamy od dawna. Celem tej funkcji jest zademonstrowanie prostoty obsługi tablic tekstowych, zdefiniowanych z określeniem znaku *NULL* jako sygnalizatora końca tekstu. Mamy tu do czynienia z tzw. *ciagiem elementów z wartownikiem* – elementem o specjalnej wartości, użytym w celu oznakowania ciągu użytecznych danych. Mianowicie znak o kodzie o wartości zero, po konwersji wartości liczbowej na wartość logiczną (wynikiem jest *fałsz*), zatrzymuje pętlę, przeglądającą tablicę tekstową znak po znaku. W funkcji *wypisz* znajduje się również przykład inicjalizacji zmiennej, nie będącej tablicą, poprzez podanie wartości inicjalizującej po znaku równości.

W obu programach pojawiło się przekazywanie tablic jako argumentów funkcji. W jaki sposób przekazać tablicę jako argument funkcji?

TABLICĘ, JAKO ARGUMENT FUNKCJI, PRZEKAZUJE SIĘ,
PODAJĄC JEJ NAZWĘ
(BEZ NAWIASÓW KWADRATOWYCH!)

Proszę spojrzeć na instrukcje wywoływania funkcji w obu omówianych listingach, operujących na tablicach – jako argument funkcji, będącej tablicą, pojawia się nazwa przekazywanej tablicy.

Na tym zakończę omawianie podstaw obsługi tablic w języku C++. Proszę rozważyć trzy problemy:

1. Po co jest potrzebny argument *długosc* funkcji *wypisz* w programie *tablice.cpp* oraz dlaczego nie istnieje analogiczny argument w funkcji *wypisz* w programie *inicjalizacja.cpp*?
2. Jak działa funkcja *wypisz* w programie *inicjalizacja.cpp*?
3. Dlaczego konieczna jest inicjalizacja zmiennej *indeks* w podanej wyżej funkcji oraz dlaczego zmienna ta jest inicjalizowana wartością zero?