Programowanie strukturalne – ćwiczenia

Tablice jednowymiarowe. Wskaźniki

1. Definicja tablicy jednowymiarowej
2. Stałe tablice
3. Tworzenie tablicy przez stałą DEFINE
4. Inicjalizacja tablicy.
5. Oznaczona inicjalizacja – od C99
6. Przypisanie wartości tablicy
7. Zakres tablicy
8. Określenie rozmiaru tablicy
9. Wskaźniki do tablic
10. Przekazywanie tablic do funkcji
11. Działania na wskaźnikach

Poniższe przykłady są z ksiżki

Richard Reese, Wskaźniki w języku C, Wydawnictwo Helion2014.

Tablica jednowymiarowa jest strukturą liniową. W celu uzyskania dostępu do elementów takiej tablicy stosuje się pojedynczy indeks. Poniżej przedstawiono deklarację pięcioelementowej tablicy na obiekty typu integer.

```
int vector[5];
```

Indeksy tablic zaczynają się od 0, a kończą na liczbie o jeden mniejszej od rozmiaru tablicy. Prawidłowe indeksy tablicy vector zaczynają się od 0, a kończą na 4. Jednakże język C nie wymusza przestrzegania tych granic.

Zastosowanie nieprawidłowego indeksu elementu tablicy może doprowadzić do nieprzewidywalnego działania aplikacji.

Wewnętrzna reprezentacja tablicy nie zawiera żadnych informacji na temat liczby elementów w tablicy. Nazwa tablicy odnosi się po prostu do bloku pamięci. Zastosowanie operatora sizeof z tablicą zwróci liczbę bajtów alokowaną dla tablicy. Aby określić liczbę elementów tablicy, należy podzielić rozmiar tablicy przez rozmiar pojedynczego elementu tablicy.

Poniżej pokazano przykład takiego działania. W wyniku uruchomienia poniższego kodu zostanie nam wyświetlona liczba 5.

```
printf("%d\n", sizeof(vector)/sizeof(int));
```

Tablica jednowymiarowa może być zainicjalizowana za pomocą instrukcji typu blokowego. W poniższej sekwencji kodu każdy element tablicy jest zainicjalizowany kolejną liczbą całkowitą, zaczynając od 1:

```
int vector[5] = {1, 2, 3, 4, 5};
```

Notacja wskaźnikowa i tablice

Wskaźniki są bardzo przydatne podczas pracy z tablicami. Mogą być stosowane do obsługi już istniejącej tablicy lub do alokacji pamięci na stercie w celu korzystania z tej alokowanej pamięci jak z tablicy. W niektórych przypadkach notacja wskaźnika i tablicy może być stosowana zamiennie, jednakże warto pamiętać, że nie są to elementy tożsame.

Gdy użyjesz samej nazwy tablicy, zwrócony zostanie jej adres. Możesz go przypisać do wskaźnika tak, jak pokazano to poniżej:

```
int vector[5] = {1, 2, 3, 4, 5};

int *pv = vector;
```

Zmienna pv nie jest wskaźnikiem na tablicę. Jest ona wskaźnikiem na pierwszy element tablicy. Przypisując wartość do zmiennej pv, przypisujemy tę wartość pierwszemu elementowi tablicy.

Możemy skorzystać z samej nazwy tablicy lub z operatora adresowania z pierwszym elementem tablicy, co pokazano poniżej. Obie czynności są równoważne i doprowadzą do zwrócenia adresu zmiennej vector. Korzystanie z operatora adresowania jest bardziej rozwlekłe, jednakże równocześnie bardziej czytelne.

```
printf("%p\n",vector);

printf("%p\n",&vector[0]);
```

Wyrażenie &vector może być czasem stosowane do uzyskania adresu tablicy. Zapis ten różni się od innych tym, że zwraca wskaźnik do całej tablicy.

Pozostałe dwa sposoby generują wskaźnik na obiekt typu integer. Wyrażenie to zamiast zwracać wskaźnik na integer, zwraca wskaźnik na tablicę obiektów typu integer.

Indeksy tablicy mogą być również stosowane ze wskaźnikami. W związku z tym zapis pv[i] jest równoznaczny z zapisem:

*(pv + i)

Wskaźnik pv zawiera adres bloku pamięci. Zapis z wykorzystaniem nawiasów odczyta adres zawarty w pv i doda go do wartości zawartej w indeksie i.

Działanie to będzie przeprowadzone na zasadach arytmetyki wskaźnikowej. Nowy adres w celu odczytania zawartości jest następnie wyłuskiwany.

Dodanie liczby całkowitej do wskaźnika doprowadzi do inkrementacji adresu przechowywanego przez wskaźnik o iloczyn rzeczonej liczby oraz rozmiaru typu danych.

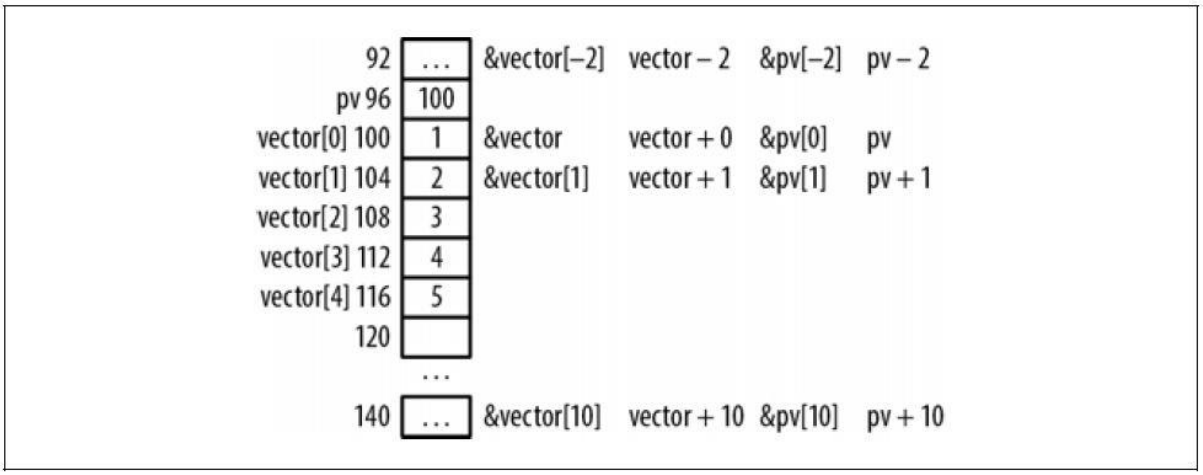
Ten sam wynik otrzymamy po dodaniu liczby całkowitej do nazwy tablicy. Poniższe zapisy są równoważne:

*(pv + i)

*(vector + i)

Załóżmy, że vector znajduje się pod adresem o numerze 100, a pv znajduje się pod adresem o numerze 96. Tabela oraz rysunek poniżej ilustrują zastosowanie indeksów tabeli i arytmetyki wskaźnikowej zarówno z nazwą tablicy, jak i wskaźnikami na różne wartości.

Wartość	**Równoważne wyrażenia**			
92	&vector[2]	vector-2	&pv[-2]	pv - 2
100	vector	vector+0	&pv[0]	pv
100	&vector[0]	vector+0	&pv[0]	pv
104	&vector[1]	vector + 1	&pv[1]	pv+1
140	&vector[10]	vector + 10	&pv[10]	pv + 10



Dodając 1 do adresu tablicy, efektywnie dodajemy do niego 4 (rozmiar elementu typu integer), ponieważ jest to tablica elementów typu integer.

Za pomocą pierwszej i ostatniej operacji adresowaliśmy obszary pamięci znajdujące się poza granicami tablicy. Nie jest to dobra praktyka, aczkolwiek pokazała Ci, że należy być ostrożnym podczas korzystania z indeksów lub wskaźników w celu uzyskania dostępu do elementów tablicy. Notacja tablicowa może być rozumiana jako operacja „przejdź i wyłuskaj”. Wyrażenie vector[2] można przetłumaczyć jako następującą instrukcję: rozpocznij od obiektu vector będącego wskaźnikiem na początek tablicy, przejdź o dwie pozycje w prawo, a na koniec wyłuskaj to miejsce w celu uzyskania wartości.

Zastosowanie operatora adresowego w połączeniu z notacją tablicową — &vector[2] — spowoduje anulowanie dereferencji. Zapis ten można interpretować jako następującą instrukcję: przejdź o dwa miejsca w lewo, a następnie zwróć wskazany adres.

Poniższy przykład przedstawia zastosowanie wskaźników w implementacji operacji dodawania skalarów. W operacji tej dana wartość mnożona jest przez każdy element wektora.

```
pv = vector;

int value = 3;

for(int i=0; i<5; i++) {

*pv++ *= value;

}
```

Różnice pomiędzy tablicami a wskaźnikami

Istnieje kilka różnic pomiędzy zastosowaniem tablic a wskaźników na tablice. Teraz będziemy korzystać z tablicy vector oraz wskaźnika pv zdefiniowanych w następujący sposób:

```
int vector[5] = {1, 2, 3, 4, 5};

int *pv = vector;
```

Kod wygenerowany przez notację vector[i] jest inny od kodu wygenerowanego przez vector+i. Notacja vector[i] generuje kod maszynowy, który rozpoczyna działanie w miejscu, w którym znajduje się tablica vector, przechodzi o i pozycji z tego miejsca i korzysta z napotkanej zawartości. Notacja vector+i generuje kod maszynowy, który rozpoczyna działanie w miejscu, w którym znajduje się tablica vector, dodaje i do adresu, a następnie korzysta z zawartości znajdującej się pod otrzymanym w ten sposób adresem.

Teoretycznie rezultat obu tych działań jest taki sam, jednak ich kod maszynowy jest różny. Różnica ta jest bez znaczenia dla większości programistów. Operator sizeof działa inaczej na tablicę, a inaczej na wskaźnik na tę samą tablicę. Operator sizeof po zastosowaniu z tablicą vector zwróci 20 — liczbę bajtów alokowanych dla tej tablicy. Operator sizeof po zastosowaniu ze wskaźnikiem pv zwróci 4 — rozmiar wskaźnika.

Wskaźnik pv jest wartością lewostronną (z ang. lvalue). Pojęcie lewej strony odnosi się do lewej strony operatora przypisania. Obiekt będący wartością lewostronną musi być modyfikowalny. Nazwa tablicy, np. vector, nie jest wartością lewostronną i nie może być modyfikowana. Adres przypisany do tablicy nie może być zmieniony. Wskaźnikowi można przypisać nową wartość. W ten sposób będzie on wskazywał na inny obszar pamięci.

Przeanalizuj poniższy przykład:

```
pv = pv + 1;

vector = vector + 1; // błąd składni
```

Nie można modyfikować tablicy vector. Możliwa jest jedynie modyfikacja jej zawartości. Jednakże, co pokazano poniżej, wyrażenie vector+1 jest poprawne:

```
pv = vector + 1;
```

Przekazywanie tablicy jednowymiarowej

Podczas przekazywania tablicy jednowymiarowej do funkcji adres tablicy jest przekazywany za pomocą wartości. Ułatwia to proces przekazywania danych, ponieważ nie musimy przekazywać całej tablicy i dokonywać dodatkowej alokacji pamięci na stosie programu.

Niestety, taki sposób przekazywania tablicy wymusza konieczność przekazania informacji określającej rozmiar tablicy. Jeżeli tego nie zrobimy, funkcja będzie posiadać tylko adres tablicy o nieokreślonym rozmiarze.

Jeżeli tablica nie zawiera żadnego integralnego elementu określającego jej rozmiar, musimy przekazać informację o rozmiarze tablicy w momencie przekazania jej adresu. W przypadku łańcucha znaków przechowywanego przez tablicę w celu określenia jego końca możemy wykorzystać znak kończący NUL. Jednakże w większości przypadków, jeżeli nie znamy rozmiaru tablicy, nie możemy przetwarzać jej elementów. Groziłoby to pominięciem części danych zawartych w tablicy lub potraktowaniem obszaru pamięci znajdującego się poza tablicą tak, jakby był jej częścią. Takie działanie zwykle doprowadza do nieprawidłowego zakończenia działania programu.

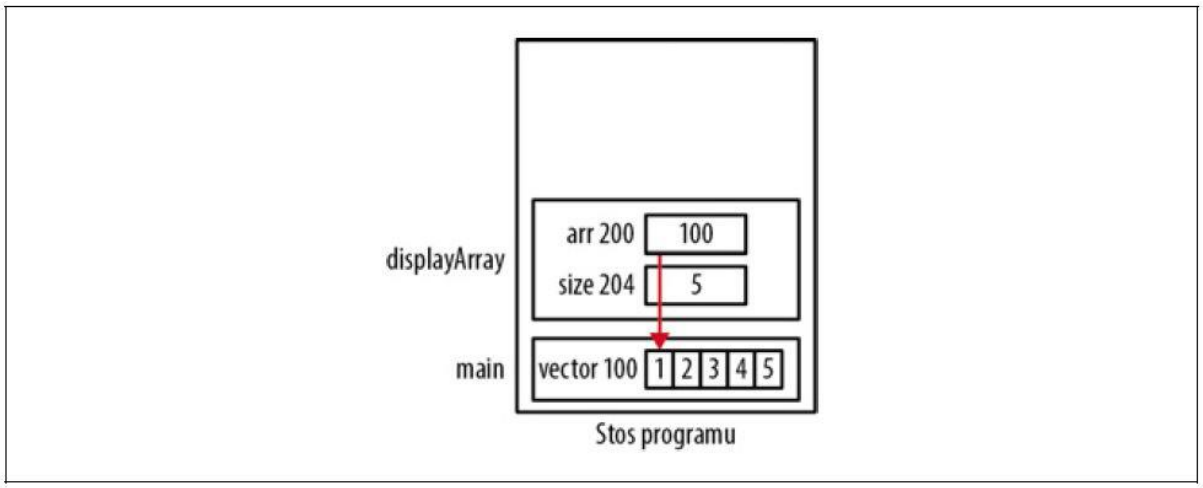
Tablicę wewnątrz funkcji możemy zadeklarować za pomocą notacji tablicowej lub notacji wskaźnikowej.

Stosowanie notacji tablicowej

W poniższym przykładzie tablicę elementów typu integer, wraz z jej rozmiarem, przekazano do funkcji. Zawartość tablicy jest następnie wyświetlana na ekranie.

```
void displayArray(int arr[], int size) {

for (int i = 0; i < size; i++) {

printf("%d\n", arr[i]);

}

}

int vector[5] = {1, 2, 3, 4, 5};

displayArray(vector, 5);
```

W wyniku działania powyższej aplikacji na ekranie zostaną wyświetlone liczby od 1 do 5. Wskazaliśmy rozmiar tablicy, przekazując do funkcji liczbę 5. Mogliśmy przekazać każdą inną liczbę dodatnią, a funkcja próbowałaby wyświetlić adekwatną liczbę elementów, niezależnie od tego, czy podana liczba odpowiadałaby prawdziwemu rozmiarowi tablicy. Gdybyśmy próbowali adresować pamięć znajdującą się poza obszarem tablicy, moglibyśmy doprowadzić do przerwania działania programu. Rysunek poniżej ilustruje alokację pamięci w omawianym przykładzie.



Często popełnianym błędem jest stosowanie operatora sizeof w celu określenia liczby elementów tablicy. Przykład takiego błędnego zapisu operacji określenia rozmiaru tablicy zaprezentowano poniżej.

W omawianym przypadku przekazalibyśmy do tablicy wartość 20.

displayArray(arr, sizeof(arr));

Częstą praktyką jest przekazywanie rozmiaru mniejszego od właściwego rozmiaru tablicy. Praktykę taką możesz stosować w celu przetwarzania tylko części tablicy. Wyobraź sobie sytuację, w której do tablicy wczytano ciąg liczb, jednakże nie wypełnia on całej tablicy. Wywołując funkcję sort, wykonałbyś sortowanie tylko wczytanych liczb, a nie wszystkich elementów tablicy.

Stosowanie notacji wskaźnikowej

Podczas zapisu deklaracji tablicowego parametru funkcji nie musisz stosować nawiasów. Zamiast notacji zawierającej nawiasy możemy zastosować notację wskaźnikową w następującej postaci:

void displayArray(int* arr, int size) {

for (int i = 0; i < size; i++) {

printf("%d\n", arr[i]);

}

}

W dalszym ciągu stosujemy notację tablicową wewnątrz funkcji. Gdybyśmy chcieli, moglibyśmy zastosować w funkcji notację wskaźnikową:

```
void displayArray(int* arr, int size) {

for (int i = 0; i < size; i++) {

printf("%d\n", *(arr+i));

}

}
```

Gdybyśmy zastosowali notację tablicową do deklarowania funkcji, to wciąż istniałaby możliwość korzystania z notacji wskaźnikowej wewnątrz funkcji:

```
void displayArray(int arr[], int size) {

for (int i = 0; i < size; i++) {

printf("%d\n", *(arr+i));

}

}
```