Programowanie strukturalne - wykład 4

dr Piotr Jastrzębski

Tablice jednowymiarowe

Definicja

Tablica - ciąg elementów jednego typu.

Deklaracja:

```
int tab[4];
char tab2[3];
float tabf[5];
```

Inicjalizacja

```
int tab[4] = {1,-3,4,5};
```

Stałe tablice

```
const int tab[4] = {1,-3,4,5};
```

Tworzenie przez stałą DEFINE

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 3

int main()
{
    int tab[ROZMIAR] = {3,-4,3};
    return 0;
}
```

Brak inicjalizacji?

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 3

int main()
{
    int tab[ROZMIAR];
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Za mało wartości?

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 3

int main()
{
    int tab[ROZMIAR] ={2,3};
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Za dużo wartości?

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 2

int main()
{
    int tab[ROZMIAR] ={2,3,-2,0};
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int tab[] ={2,3,-2,0};
    for(int i=0;i<sizeof tab/sizeof(int);i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Oznaczona inicjalizacja - od C99

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 5

int main()
{
    int tab[ROZMIAR] = {[2] = 34};
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Przypisanie wartości

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 5

int main()
{
    int tab[ROZMIAR] = {[2] = 34};
    tab[0]=2;
    for(int i=0;i<ROZMIAR;i++)
    {
        printf("%d\n",tab[i]);
    }
    return 0;
}
```

Zakres tablic

```c
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 5

int main()
{
    int tab[ROZMIAR] = {1,-2,3};
    printf("%d",tab[-1]);
    return 0;
}
```

Jak określić rozmiar inaczej niż przez stałą DEFINE?

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n = 5;
    int m = 8;
    int tabb1[5]; // ok
    int tabb2[3*2+1]; // ok
    int tabb3[sizeof(int)+1]; // ok
    //int tabb4[-2]; // nie,  bo <0
    int tabb5[0]; // nie ???
    //int tabb6[6.5]; // nie, boe nie całk.
    //int tabb7[int(2.5)]; // ok, mamy rzut
    int tabb8[n]; // brak dla C90
    int tabb9[m]; // Brak dla C()
    return 0;
}
```

Wskaźniki do tablic

Nazwa tablicy jest równocześnie adresem jej pierwszego elementu.

```
#include <stdio.h>
#include <stdlib.h>
#define ROZMIAR 3

int main()
{
    int tab[ROZMIAR] = {4,0,-2};
    printf("%i",tab==&tab[0]);
    return 0;
}
```

Równoważne:

```
tab + 3 == &tab[3] // ten sam adres
*(tab + 3) == tab[2] // ta sama wartosc
```

Uwaga na nawiasy:

```
*(tab +2) // wartosc elementu tablicy o indeksie 2
*tab +2 // 2 dodane do wartosci elementu o indeksie 0
```

Przekazywanie tablic do funkcji

Równoważnie

```
int suma(int *tab, int n)
{
// tutaj kod
}
```

```
int suma(int tab[], int n)
{
// tutaj kod
}
```

Uwaga: nie możemy wprost odczytać rozmiaru(!).

Działania na wskaźnikach

- Przypisanie - przypisywana wartość może być tablicą albo zmienną poprzedzoną operatorem adresu (&), ewentualnie innym wskaźnikiem.

```
#include <stdio.h>
int main(void)
{
    int tab[4]={1,-3,4,2};
    int *wsk1, *wsk2;
    wsk1=tab;
    wsk2=&tab[2];
}
```

- Pobranie wartości (dereferencja, wyłuskiwanie) - *
- Pobranie adresu wskaźnika - &

- Dodawanie liczb całkowitych do wskaźnika — za pomocą operatora + można dodać liczbę całkowitą do wskaźnika albo wskaźnik do liczby całkowitej. Wtedy liczba całkowita jest mnożona przez liczbę bajtów zajmowaną przez wartość wskazywanego typu, a wynik jest dodawany do pierwotnej wartości adresu.

```
#include <stdio.h>
int main(void)
{
    int tab[4]={1,-3,4,7};
    int *wsk1, *wsk2;
    wsk1=tab;
    wsk2=wsk1+3;
    printf("%p %p\n",wsk1,wsk2);
    printf("%d %d\n",*wsk1,*wsk2);
    return 0;
}
```

- Zwiększenie wskaźnika (inkrementacja). Można je uzyskać przez zwykłe dodawanie lub za pomocą operatora inkrementacji. Zwiększenie wskaźnika do elementu tablicy sprawia, iż wskazuje on kolejny element.

```
#include <stdio.h>
int main(void)
{
    int tab[4]={1,-3,4,7};
    int *wsk1, *wsk2;
    wsk1=tab;
    printf("%p %d\n",wsk1,*wsk1);
    wsk1++;
    printf("%p %d\n",wsk1,*wsk1);
    return 0;
}
```

- Odejmowanie liczby całkowitej od wskaźnika — w tym celu można użyć operatora - ; wskaźnik musi być pierwszym z operandów. Liczba całkowita jest mnożona przez liczbę bajtów zajmowaną przez wartość wskazywanego typu, a wynik jest odejmowany od pierwotnego adresu.
- Zmniejszenie (dekrementacja) wskaźników - analogicznie.

- Odejmowanie. Możliwe jest znalezienie różnicy między dwoma wskaźnikami. Działanie to jest wykonywane zwykle na wskaźnikach do elementów tej samej tablicy — w celu określenia, jak daleko od siebie się znajdują. Wynik jest wyrażony w jednostce o rozmiarze typu.

```c
#include <stdio.h>
int main(void)
{
    int tab[4]={1,-3,4,7};
    int *wsk1, *wsk2;
    wsk1=tab;
    wsk2=&tab[3];
    printf("%d\n",wsk2-wsk1);
    return 0;
}
```

- Porównanie — możemy używać operatorów relacyjnych, aby porównać wartości dwóch wskaźników tego samego typu.

```c
#include <stdio.h>
int main(void)
{
    int tab[4]={1,-3,4,7};
    int *wsk1, *wsk2;
    wsk1=tab;
    wsk2=&tab[3];
    printf("%d\n",wsk2>wsk1);
    return 0;
}
```

Duży błąd merytoryczny - 2 na egzaminie(!)

Dereferencja niezainicjalizowanych wskaźników:

```c
#include <stdio.h>
int main(void)
{
    int *wsk; // niezainicjalizowany wskaznik
    *wsk = 5;
    return 0;
}
```

Bibliografia

- Stephen Prata, Język C. Szkoła programowania. Wydanie VI, Wyd. Helion, 2016.
- https://pl.wikibooks.org/wiki/C/Tablice, dostęp online 20.03.2020.
- https://pl.wikibooks.org/wiki/C/Tablice_-_wi%C4%99cej, dostęp online 20.03.2020.