Pracownia programowania - Wykład 10

Tablice “wielowymiarowe”

Misz-masz definicji

Tablice wielowymiarowe (elementów)

- tablice wielowymiarowe o stałym rozmiarze (?)
- tablice wielowymiarowe statyczne (?)

Tablice tablic

- tablice wielowymiarowe dynamiczne (?)
- tablice wielowymiarowe o zmiennym rozmiarze (?)

Tablice wielowymiarowe “statyczne”

Tablice wielowymiarowe “statyczne”

Cechy:

- stały rozmiar, niezmienny w trakcie działania programu
- by użyć - musi być zadeklarowana

Deklaracja

Składnia

```
typ nazwa[ wymiar1 ][ wymiar2 ]...[ wymiarN ];
```

Przykład:

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

int main()
{
    int tab[2][3];
    return 0;
}
```

Inicjalizacja

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

int main()
{
    int tab[2][3] = {{1,2,4},{-2,3,5}};
    return 0;
}
```

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

int main()
{
    int tab[2][3];
    tab[0][0]=1;
    tab[0][1]=2;
    tab[0][2]=4;
    tab[1][0]=-2;
    tab[1][1]=3;
    tab[1][2]=5;
    return 0;
}
```

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

int main()
{
    int tab[2][4] = {{1,2,4}};
    return 0;
}
```

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

int main()
{
    int tab[2][3] = {{1,2,4},{-2,3,5}};
    printf("%p\n",&tab[0][0]);
    printf("%p\n",&tab[0][1]);
    printf("%p\n",&tab[0][2]);
    printf("%p\n",&tab[0][3]);
    printf("%p\n",&tab[1][0]);
    printf("%p\n",&tab[1][1]);
    printf("%p\n",&tab[1][2]);
    return 0;
}
```

Przekazanie tablicy do funkcji

```c
void foo1(int n, int m, int tab[n][m])
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            printf("[%d,%d]=%d ",i,j,tab[i][j]);
        }
        printf("\n");
    }
}
int main()
{
    int tab[2][3] = {{1,2,4},{-2,3,5}};
    foo1(2,3,tab);
    return 0;
}
```

```c
void foo2(int n, int m, int tab[][m])
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            printf("[%d,%d]=%d ",i,j,tab[i][j]);
        }
        printf("\n");
    }
}
int main()
{
    int tab[2][3] = {{1,2,4},{-2,3,5}};
    foo2(2,3,tab);
    return 0;
}
```

```c
void foo3(int tab[2][3])
{
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<3;j++)
        {
            printf("[%d,%d]=%d ",i,j,tab[i][j]);
        }
        printf("\n");
    }
}
int main()
{
    int tab[2][3] = {{1,2,4},{-2,3,5}};
    foo3(tab);
    return 0;
}
```

```c
void foo4(int tab[][3])
{
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<3;j++)
        {
            printf("[%d,%d]=%d ",i,j,tab[i][j]);
        }
        printf("\n");
    }
}
int main()
{
    int tab[2][3] = {{1,2,4},{-2,3,5}};
    foo4(tab);
    return 0;
}
```

Mieszamy ze wskaźnikami

Równoważnie

```
tab[i][j]
*(*(tab+i)+j)
```

Tablice “dynamiczne”

Tablice tablic

Tablice “dynamiczne” / Tablice tablic

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

int main()
{
    int **tab = (int**) malloc(sizeof(int*)*2);
    tab[0]=(int*) malloc(sizeof(int)*3);
    tab[1]=(int*) malloc(sizeof(int)*3);
    return 0;
}
```

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

int main()
{
    int **tab = (int**) malloc(sizeof(int*)*2);
    tab[0]=(int*) malloc(sizeof(int)*3);
    tab[1]=(int*) malloc(sizeof(int)*3);
    free(tab[0]);
    free(tab[1]);
    free(tab);
    return 0;
}
```

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

void foo(int **tab, int n, int m)
{

}

int main()
{
    int **tab = (int**) malloc(sizeof(int*)*2);
    tab[0]=(int*) malloc(sizeof(int)*3);
    tab[1]=(int*) malloc(sizeof(int)*3);
    foo(tab,2,3);
    return 0;
}
```

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

int ** foo(int n, int m)
{
    int **tab = (int**) malloc(sizeof(int*)*n);
    tab[0]=(int*) malloc(sizeof(int)*m);
    tab[1]=(int*) malloc(sizeof(int)*m);
    return tab;
}

int main()
{
    int **t=foo(2,3);
    return 0;
}
```

Porównanie

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

int main()
{
    int **tab = (int**) malloc(sizeof(int*)*2);
    tab[0]=(int*) malloc(sizeof(int)*3);
    tab[1]=(int*) malloc(sizeof(int)*3);
    *(*(tab+0)+0) = 8;
    *(*(tab+0)+1) = -2;
    *(*(tab+0)+2) = 4;
    *(*(tab+1)+0) = 6;
    *(*(tab+1)+1) = 7;
    *(*(tab+1)+2) = -4;
    for(int i=0;i<2;i++){
        printf("%p\n", &tab[i]);
        printf("%p\n", tab[i]);
        for(int j=0;j<3;j++){
```

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

int main()
{
    int tab[2][3] = {{8,-2,4},{6,7,-4}};
    for(int i=0;i<2;i++){
        printf("%p\n", &tab[i]);
        printf("%p\n", tab[i]);
        for(int j=0;j<3;j++){

            printf("[%d,%d]=%d, adres = %p\n",i,j,tab[i][j], &tab[i][j]);

        }
    }
    return 0;
}
```

Skomplikujmy to jeszcze bardziej

Tak!

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

int main()
{
    int tab[3][2] = {{2,-3},{5,4},{8,7}};
    //wskaznik do tablicy dwóch wartosci int
    //operator [] ma wyzszy priorytet niz *
    int (*wsk)[2];
    wsk=tab;
    printf("%d %p %p\n",**wsk,*wsk,wsk);
    printf("%d %p %p\n",**(wsk+1),*(wsk+1),wsk+1);
    printf("%d %p\n",*(wsk[0]),wsk[0]);
    printf("%d %p\n",*(wsk[1]),wsk[1]);
    printf("%d %p\n",*(wsk[2]),wsk[2]);
    printf("%d %p\n",*(wsk[1]+1),wsk[1]+1);
    printf("%d %p\n",*(*wsk+1),*wsk+1);
    printf("%d\n",*wsk[1]);
```

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

int main()
{
    //tablica dwoch wskazników do int
    int *wsk[2]= {
    (int[]) {3,4,5},
    (int[]) {-2,3,-4},
    };
    printf("%d %p\n",wsk[0][0],&wsk[0][0]);
    printf("%d %p\n",wsk[0][1],&wsk[0][1]);
    printf("%d %p\n",wsk[0][2],&wsk[0][2]);
    printf("%d %p\n",wsk[0][3],&wsk[0][3]);
    printf("%d %p\n",wsk[1][0],&wsk[1][0]);
    printf("%d %p\n",*(*(wsk+1)+2),*(wsk+1)+2);
    return 0;
}
```

Tablice postrzępione

Tablica postrzępiona jest to taka dwuwymiarowa tablica, która posiada różną liczbę kolumn w każdym rzędzie.

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

int main()
{
    int (*(arr2[])) = {
        (int[]) {0, 1, 4, 3},
        (int[]) {4, -1},
        (int[]) {6, -2, 8}
        };
    return 0;
}
```

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

int main()
{
    int **tab = (int**) malloc(sizeof(int*)*2);
    tab[0]=(int*) malloc(sizeof(int)*4);
    tab[1]=(int*) malloc(sizeof(int)*3);
    return 0;
}
```

Więcej wymiarów?

```
int *** alokuj(int n, int m, int k)
{
    int *** tab = malloc(n*sizeof(int **));
    for(int i=0;i<n;i++)
    {
        tab[i] = malloc(m*sizeof(int*));
    }
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            tab[i][j] = malloc(k*sizeof(int));
        }
    }
    return tab;
}

void zwolnij(int *** tab, int n, int m, int k)
```

Bibliografia

- Stephen Prata, Jezyk C. Szkoła programowania. Wydanie VI, Wyd. Helion, 2016.
- https://cybersecurity.umcs.lublin.pl/wp-content/uploads/kmazur/PP2017/, dostęp online 1(
- Richard Reese, Wskaźniki w języku C, Wydawnictwo Helion 2014.
- http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/skp/W11_wskazniki_na_tablice_wiel(dostęp online 15.04.2023.
- https://pl.wikibooks.org/wiki/C/Typy_z%C5%82o%C5%BCone#Struktury, dostęp online 20.(