Programowanie strukturalne - wykład 9 i 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

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

```c
#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

```
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 “dynamiczne”

```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;
}
```

Kod do analizy

https://gist.github.com/pjastr/4ac29959e212436807c626610cb1b964

Skomplikujmy to jeszcze bardziej

Tak!

Kod do analizy2

https://gist.github.com/pjastr/cfa14265a177917593ee017a5e8baa3a

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;
}
```

Bibliografia

- Stephen Prata, Jezyk C. Szkoła programowania. Wydanie VI, Wyd. Helion, 2016.
- https://cybersecurity.umcs.lublin.pl/wp-content/uploads/kmazur/PP2017/, dostęp online 10.04.2020.