Programowanie strukturalne - 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:

```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

```
1  void foo1(int n, int m, int tab[n][m])
2  {
3      for(int i=0;i<n;i++)
4      {
5          for(int j=0;j<m;j++)
6          {
7              printf("[%d,%d]=%d ",i,j,tab[i][j]);
8          }
9          printf("\n");
10     }
11 }
12 int main()
13 {
14     int tab[2][3] = {{1,2,4},{-2,3,5}};
15     foo1(2,3,tab);
16     return 0;
17 }
```

```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

```
1  tab[i][j]
2  *(*(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 (1)

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

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

Kod do analizy (2)

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

int main()
{
    int tab[2][3];
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<3;j++)
        {
            printf("TAB[%d, %d]= ",i,j);
            scanf("%d",*(tab+i)+j);
            printf("%p \n",&tab[i][j]);
        }
    }
    return 0;
}
```

Kod do analizy (3)

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

int main()
{
    int tab[2][3];
    int (*wsk_w)[3]; //wskaznik na wiersz (czyli 3 elementową tablice int)
    int*wsk_k; // wskaznik na kolumne (czyli na liczbę int)
    for(wsk_w = tab;wsk_w < tab + 2;wsk_w++)
    {
        for(wsk_k = *wsk_w;wsk_k < *wsk_w + 3;wsk_k++)
        {
            printf("TAB[%Id, %Id]= ",wsk_w-tab, wsk_k-*wsk_w);
            scanf("%d",wsk_k);
            printf("%p \n",wsk_k);
        }
    }
    return 0;
}
```

Kod do analizy (4)

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

int main()
{
    int tab[2][3];
    tab[ 0 ][ 0 ] = 5;
    **tab=7;
    return 0;
}
```

Kod do analizy (5)

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

int main()
{
    int**tabD = malloc(4*sizeof(int*));
    for(int i=0;i<4;i++)  // for(int i=0;i<4;i++)
    {
        *(tabD+i) = malloc(5*sizeof(int)); // tabD[i] = malloc(5*sizeof(int
    }
    *(*(tabD+2)+3) = 111;   // tabD[2][3] = 111;
    //zamiana miejscami wierszy o indeksach 1 i 3
    int* wsk_pom;
    wsk_pom = *(tabD +1); // wsk_pom = tabD[0];
    *(tabD + 1) = *(tabD + 3); // tabD[0] = tabD[3];
    *(tabD + 3) = wsk_pom; // tabD[3] = wsk_pom;
    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
- Richard Reese, Wskaźniki w języku C, Wydawnictwo Helion 2014.
- http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/skp/W11_wskazniki_na_tablice_wielo dostęp online 15.04.2023.
- https://pl.wikibooks.org/wiki/C/Typy_z%C5%82o%C5%BCone#Struktury, dostęp online 20.0