

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №10

з дисципліни

«Алгоритмізація та програмування І»

Виконав:

студент групи КН-108

Зінько Павло

Викладач:

Гасько Р.Т.

Львів – 2018р

Варіант №13

Тема: "Динамічні масиви"

Мета: Організація динамічних масивів.

Постановка завдання

Написати програму, у якій створюються динамічні масиви й виконати їхню обробку у відповідності до свого варіанту

Виконати завдання:

13

Сформувати двовимірний масив. Знищити з нього всі стовпці, у яких зустрічається задане число.

Код:

```
#include <stdio.h>
#include <malloc.h>
#include <time.h>
#include <stdlib.h>
#include <stdbool.h>
```

```
//мітка при якій коли функція бачить її буде переривати цикл
```

```
#define POINT 999
```

```
int count;
```

```
//функція передання вказівника
```

```

int** initialize(int size1,int size2)
{
    // вказівник на вказівник на рядок елементів
    int **arr;

    // Виділення пам'яті для вказівника на рядки
    arr = malloc(size2 * sizeof(int*));

    // цикл по рядкам
    for(int i = 0; i < size2;i++)
    {
        // Виділення пам'яті для зберегання рядків
        arr[i] = malloc(size1*sizeof(int));
    }
    return arr;
}

```

// функція заповнення нашого масиву

```

void create(int** arr,int size1, int size2)
{
    srand(time(NULL));
    for(int i = 0; i < size1;i++)
    {
        for(int j = 0;j < size2;j++)
        {
            arr[j][i] = rand()%20;
        }
    }
}

```

```

        }
    }
}

```

// Вивід згенерованого масива

```

void print(int** arr,int size1,int size2)
{
    for(int i = 0;i < size1;i++)
    {
        for(int j = 0;j< size2;j++)
        {
            printf("%d\t",arr[j][i]);
        }
        printf("\n");
    }
    printf("\n");
}

```

// Функція зміни та перенесення нового стовпчика

```

void point(int** arr,int size1,int size2)
{
    //елемент який потрібно видалити
    int Element;
    printf("Give me the element for deliting:\n");
    scanf("%d",&Element);
    printf("\n");
}

```

```

for(int i = 0; i < size1;i++)
{
    for(int j = 0; j < size2; j++)
    {
        if(arr[j][i] == Element)
        {
            for(int k = 0; k < size2;k++)
            {
                // якщо найшло цю мітку записує кількість
                // стовпців які потрібно замінити
                arr[j][k] = POINT;
                if(k == size2-1)
                    count++;
            }
        }
    }
}

```

// функція видалення помічених функцією "point" стовпців

```

int** kill(int** arr,int** arr2,int size1, int size2)

```

```

{
    int tmpi = 0;
    for(int i = 0; i < size2;i++)
    {
        for(int j = 0; j < size1;j++)

```

```

        {
            // якщо найшло цю мітку,перериває цикл
            if(arr[i][j] == POINT)
            {
                break;
            }

            arr2[tmpi][j] = arr[i][j];

            //записує кількість стовпців які потрібно видалити
            if(j==size1-1)
                tmpi++;
        }
    }

    // вертає новий стовпчик
    return arr2;
}

// Звільняє пам'ять та видаляє масив
void freeArr(int** arr,int size1)
{
    for(int i = 0; i < size1;i++)
    {
        free(arr[i]);
    }
}

```

```
}
```

```
int main()
```

```
{
```

```
    int size1;
```

```
    int size2;
```

```
    printf("Print size of array\n");
```

```
    scanf("%d",&size2);
```

```
    scanf("%d",&size1);
```

```
    printf("\n");
```

```
    // створює наш масив
```

```
    int** arr = initialize(size1,size2);
```

```
    // ініціалізує наш масив
```

```
    create(arr,size1,size2);
```

```
    //виводить згенерований масив
```

```
    print(arr,size1,size2);
```

```
    point(arr,size1,size2);
```

```
    // ініціалізує наш масив але без даних стовпців
```

```
    int** arr1 = initialize(size1,size2-count);
```

```
    arr1 = kill(arr,arr1,size1,size2);
```

```
// виводить наш масив але без даних стовпців  
print(arr1,size1,size2-count);
```

```
// звільняє пам'ять  
freeArr(arr,size1);  
freeArr(arr1, size1-1);
```

```
}
```



```

1  #include <stdio.h>
2  #include <malloc.h>
3  #include <time.h>
4  #include <stdlib.h>
5  #include <stdbool.h>
6
7  //мітка при якій коли функція бачить її буде переривати цикл
8  #define POINT 999
9
10 int count;
11
12 //функція передання вказівника
13 int** initialize(int size1,int size2)
14 {
15     // вказівник на вказівник на рядок елементів
16     int **arr;
17
18     // Виділення пам'яті для вказівника на рядки
19     arr = malloc(size2 * sizeof(int*));
20
21     // цикл по рядках
22     for(int i = 0; i < size2;i++)
23     {
24         // Виділення пам'яті для збереження рядків
25         arr[i] = malloc(size1*sizeof(int));
26     }
27     return arr;
28 }
29
30 // функція заповнення нашого масиву
31 void create(int** arr,int size1, int size2)
32 {
33     srand(time(NULL));
34     for(int i = 0; i < size1;i++)
35     {
36         for(int j = 0;j < size2;j++)
37         {
38             arr[j][i] = rand()%20;
39         }
40     }
41 }
42
43 // Вивід згенерованого масива
44 void print(int** arr,int size1,int size2)
45 {
46     for(int i = 0;i < size1;i++)
47     {
48         for(int j = 0;j< size2;j++)
49         {
50             printf("%d\t",arr[j][i]);
51         }
52         printf("\n");
53     }
54     printf("\n");
55 }
56
57 // функція зміни та перенесення нового стовпчика
58 void point(int** arr,int size1,int size2)
59 {
60     //елемент який потрібно видалити
61     int Element;
62     printf("Give me the element for deliting:\n");
63     scanf("%d",&Element);
64     printf("\n");
65
66     for(int i = 0; i < size1;i++)
67     {
68         for(int j = 0; j < size2; j++)
69         {
70             if(arr[j][i] == Element)
71             {
72                 for(int k = 0; k < size2;k++)
73                 {
74                     // якщо знайшло цю мітку запише кількість стовпців які потрібно замінити

```

```

73     }
74     // якщо найшло цю мітку записує кількість стовпців які потрібно замінити
75     arr[j][k] = POINT;
76     if(k == size2-1)
77         count++;
78     }
79 }
80 }
81 }
82 }
83
84 // функція видалення помічених функцією "point" стовпців
85 int** kill(int** arr,int** arr2,int size1, int size2)
86 {
87     int tmpi = 0;
88     for(int i = 0; i < size2;i++)
89     {
90         for(int j = 0; j < size1;j++)
91         {
92             // якщо найшло цю мітку,перериває цикл
93             if(arr[i][j] == POINT)
94             {
95                 break;
96             }
97             arr2[tmpi][j] = arr[i][j];
98         }
99         //записує кількість стовпців які потрібно видалити
100         if(j==size1-1)
101             tmpi++;
102     }
103 }
104
105 // вертає новий стовпчик
106 return arr2;
107 }
108 }
109
110 // звільняє пам'ять та видалляє масив
111 void freeArr(int** arr,int size1)
112 {
113     for(int i = 0; i < size1;i++)
114     {
115         free(arr[i]);
116     }
117 }
118
119 int main()
120 {
121     int size1;
122     int size2;
123     printf("Print size of array\n");
124     scanf("%d",&size2);
125     scanf("%d",&size1);
126     printf("\n");

```

```

127
128     // створює наш масив
129     int** arr = initialize(size1,size2);
130
131     // ініціалізує наш масив
132     create(arr,size1,size2);
133
134     //виводить згенерований масив
135     print(arr,size1,size2);
136     point(arr,size1,size2);
137
138
139     // ініціалізує наш масив але без даних стовпців
140     int** arr1 = initialize(size1,size2-count);
141     arr1 = kill(arr,arr1,size1,size2);
142
143     // виводить наш масив але без даних стовпців
144     print(arr1,size1,size2-count);
145
146     // звільняє пам'ять
147     freeArr(arr,size1);
148     freeArr(arr1, size1-1);
149 }
150 }

```

Результат:

```
~/workspace/ $ make laba10
clang -fsanitize=signed-integer-over
~/workspace/ $ ./laba10
Print size of array
4
4

14      19      8      6
3       10      5     18
13      5       4     18
14      1      14      6

Give me the element for deliting:
5

14      6
3       18
13      18
14      6
```