Повторення

Функції

Програмні модулі мови с

- Найкращим засобом розробки програми і підтримки великих програм є конструювання програми у вигляді невеликих окремих частин – модулів.
- Модулі мови С називаються ***функціями***. Програми на С розроблюються, як правило, шляхом об'єднання нових функцій, які розроблені програмістом, із функціями, які поставляються у складі системної бібліотеки мови С.

Звертання до функції називають **викликом функції.** У виклику функції вказується її ім'я та передається інформація (як аргументи), яка необхідна для виконання функції.

Після виконання функції програма повертається в те місце, звідки відбувався виклик функції. Виклик функції може бути записаний як у головній функції (main), так і в будь-якій іншій функції.

Функції дозволяють розбити програму на модулі.

Усі змінні, які об'явлені в тілі функції, є ***локальними змінними*** – вони відомі тільки функції, в якій вони визначені.

Більшість функцій мають ***список параметрів***. Параметри дозволяють функціям обмінюватися інформацією. Параметри функції – це також локальні змінні.

Визначення функції

тип_поверненого_значення ім'я_функції (список_параметрів)

{

об'яви

оператори

}

Приклади об’яви функцій

```
int Fun1 (int a, float b, int c);
char Fun2 (int x, char y);
void Fun3 (char h, int  n, float tmp);
float Fun4 (float q, int t, float r, int k) ;
```

Як ім'я функції може бути будь-який допустимий ідентифікатор. Типом результату, який повертає функція, є ***тип_поверненого_значення***. Якщо як тип задано ключове слово **void**, це означає, що функція не повертає **нічого**.

Якщо ***тип_поверненого_значення*** не вказаний, компілятор вважає, що тип має значення **int**.

Список_параметрів - це список об'яв параметрів (відокремлених комами), які отримує функція в момент її виклику. Якщо функція не отримує значень, ***список_параметрів*** позначається ключовим словом **void**. Тип кожного параметра повинен бути описаний, за виключенням типу **int**. Якщо тип не вказаний, вважається, що параметр має тип **int**.

- ***Об'яви*** та ***оператори*** у середині фігурних дужок складають ***тіло функції***.

Перед першим викликом функція повинна бути визначена

- або повністю описана перед функцією main()
- або за допомогою ***прототипу***.

Компілятор використовує прототип функції для перевірки того, що - виклик функції має коректний тип поверненого значення,

- коректне число аргументів,
- коректний тип аргументів і
- коректний порядок слідування аргументів.

Існують способи повернення управління у ту точку програми, в якій була викликана функція:

- якщо функція не повертає результат, управління повертається, як тільки зустрічається права фігурна дужка, що завершує тіло функції,

- якщо функція повертає результат, тоді оператор

return <u>вираз</u>;

повертає значення ***<u>виразу</u>***.

Приклад 1. Написати функцію, що знаходить добуток 3 чисел

```
#include <stdio.h>
#include <stdlib.h>
 float fun1(float q, float w, float e);  //прототип функції
int main()
{float a,s,d;
int n;
    printf("Hello!\n");
```

```
do
    {
    printf("Input 3 numbers\n");
    scanf ("%f%f%f", &a,&s,&d);
printf("\n\t\tresult = %f\n", fun1(a,s,d)); // виклик функції
     n=(int)getch();
    }
    while (n!=27);
    return 0;
}
```

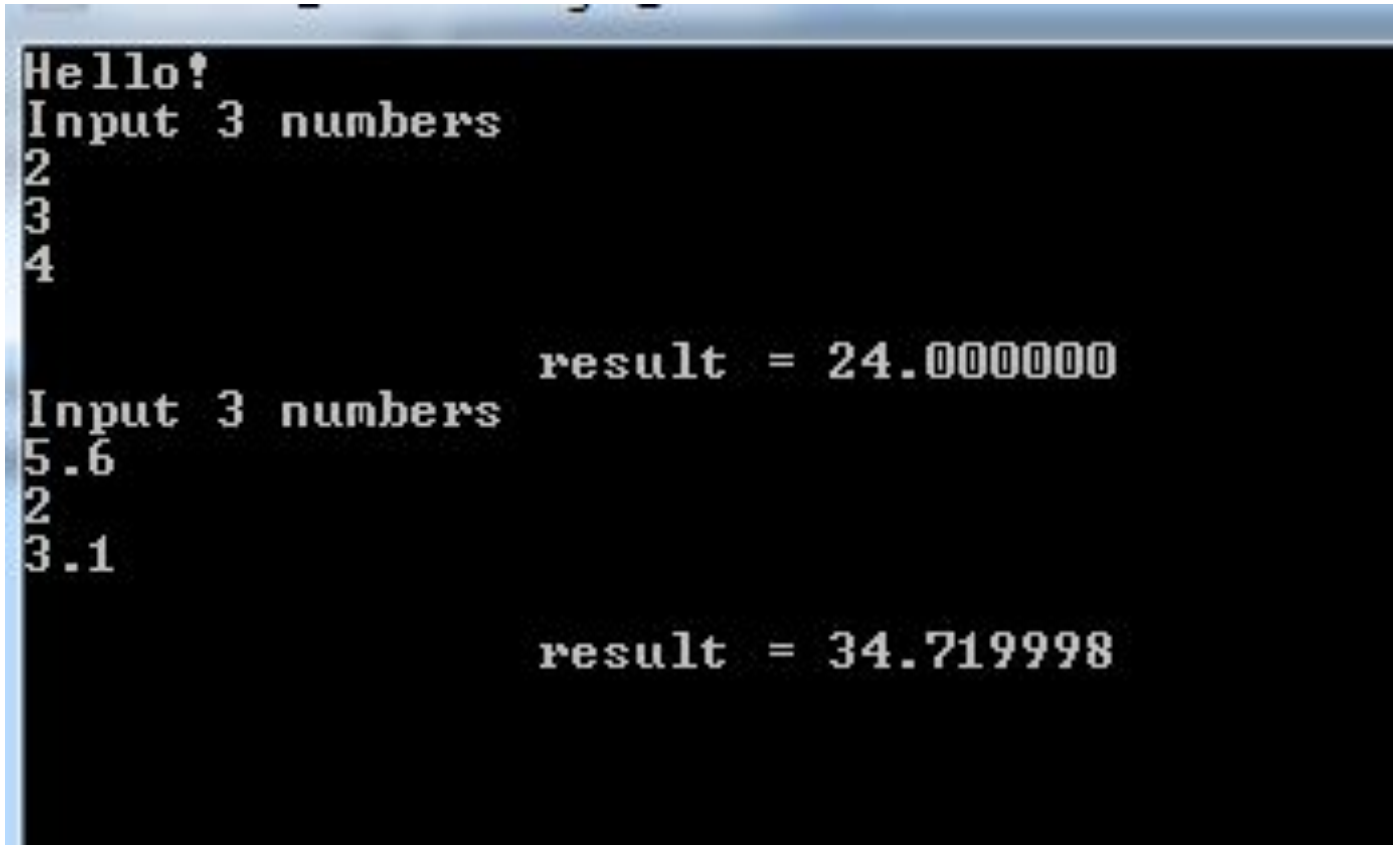
```
//опис функції
float fun1(float q, float w, float e)
{
    return q*w*e;
}
```

Приклад 1а. Написати функцію, що знаходить добуток 3 чисел (альтернатива)

```
#include <stdio.h>
#include <stdlib.h>
//опис функції
 float fun1(float q, float w, float e)
{ float z;
z= q*w*e;
    return z;
}
```

```
int main()
{float a,s,d,f;
int n;
   printf("Hello!\n");
do
    {
    printf("Input 3 numbers\n");
    scanf ("%f", &a);
scanf ("%f", &s);
scanf ("%f", &d);
```

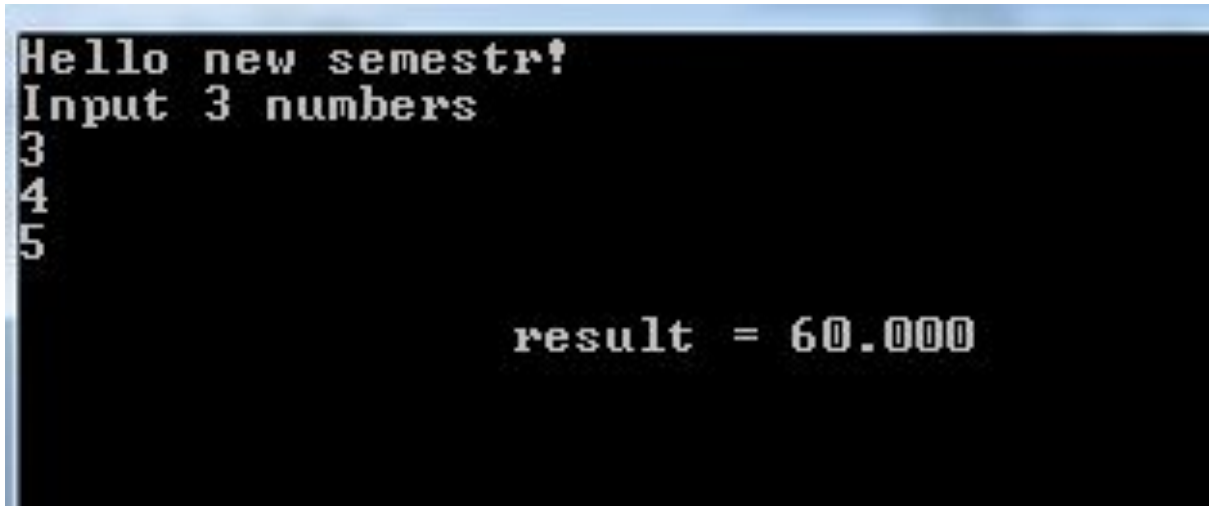
```
f= fun1(a,s,d);// виклик функції
printf("\n\t\tresult = %f\n", f);
     n=(int)getch();
    }
    while (n!=27);
    return 0;
}
```


Hello!
Input 3 numbers
2
3
4
result = 24.000000
Input 3 numbers
5.6
2
3.1
result = 34.719998

Приклад 1б. Написати функцію, що знаходить добуток 3 чисел

```
#include <stdio.h>
#include <stdlib.h>
 float fun1(float q, float w, float e);

 void HH(void)
 {
    printf("Hello new semestr!\n");
 }
```

```
int main()
{float a,s,d,f,g;
int n;
HH();
 do
    {
    printf("Input 3 numbers\n");
    scanf ("%f%f%f", &a,&s,&d);
printf("\n\t\tresult = %.3f\n", fun1(a,s,d));
     n=(int)getch();
    }
    while (n!=27);
    return 0;}
```


Hello new semestr!
Input 3 numbers
3
4
5
result = 60.000

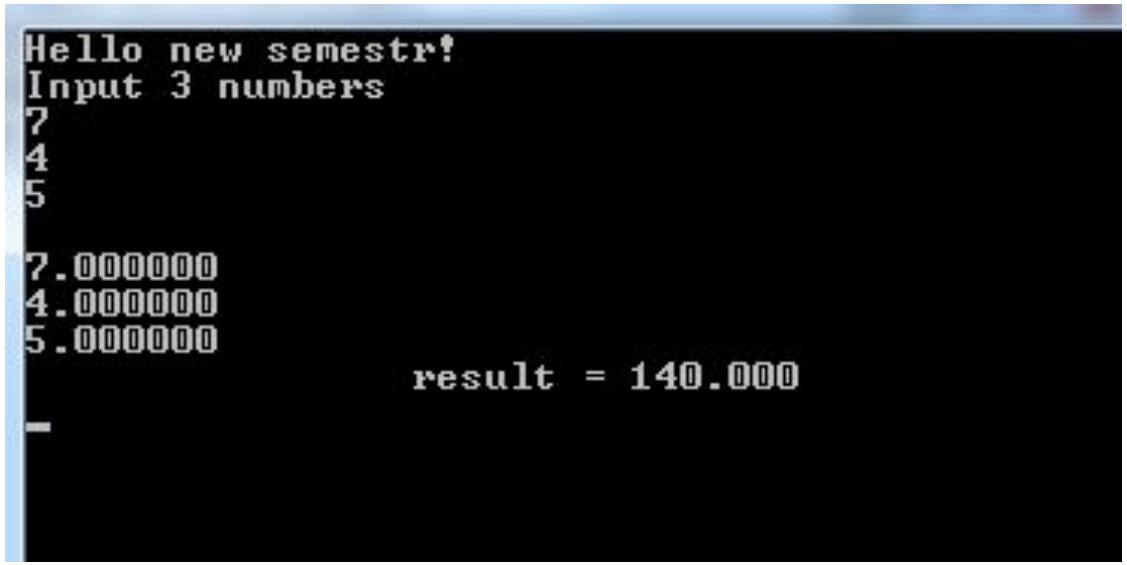
Приклад 1в. Написати функцію, що знаходить добуток 3 чисел

```
#include <stdio.h>
#include <stdlib.h>
void output_number(float z,float x, float c)
{
    printf("\t\n%f", z);
    printf("\t\n%f", x);
    printf("\t\n%f", c);
}
```

```
float fun1(float q, float w, float e);

 void HH(void)
 {
    printf("Hello new semestr!\n");
 }
```

```
int main()
{float a,s,d,f,g;
int n;
HH();
    do
   {
   printf("Input 3 numbers\n");
   scanf ("%f%f%f", &a,&s,&d);
   output_number(a,s,d);
    g= fun1(a,s,d));
    printf("\n\t\tresult = %.3f\n", g);
    n=(int)getch();
   }
   while (n!=27);
   return 0;}
```

```
float fun1(float q, float w, float e)
{ float t;
t= q*w*e;
    return t;
}
```


Hello new semestr!
Input 3 numbers
7
4
5
7.000000
4.000000
5.000000
result = 140.000

Масиви

Масив є групою комірок пам'яті, які мають одне і те ж саме ім'я та однаковий тип.

Для використання конкретної комірки або елементу масиву вказується **ім'я масиву та зміщення цієї комірки відносно першої комірки або початку масиву.**

Зміщення вказується після імені масиву у квадратних дужках і називається ***індексом*** масиву.

У мові С індекси починаються з **0**

int Q[10];

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]	Q[5]	Q[6]	Q[7]	Q[8]	Q[9]

Об’ява одновимірного масиву

```
int Arr[10];
float  B[200];
char RRR[15];
```

Об’ява двовимірного масиву

```
char T_3 [13][2];
float  B2[3][147];
int A1[10][10];
```

Приклад 2. Дано три масиви дійсних чисел A[8], B[6], C[3]. Нормувати елементи кожного масиву по максимальному(тобто розділити всі елементи масиву на його максимальний елемент)

```
#include <stdio.h>
#include <stdlib.h>
#define R 8
#define T 6
#define Y 3

void modifyArray(int N,float X[N]);
 void InputArray(int N,float X[N]);
 void OutArray(int N,float X[N]);
```

```
int main ()
{ int w;
float A[R];
float B[T];
float C[Y];
 printf ("Input quantity of elements of Array A \n");
 scanf("%d",&w);
 printf ("Input Array A \n");
 InputArray(w,A);
OutArray(w,A);
modifyArray(w,A);
getch();
```

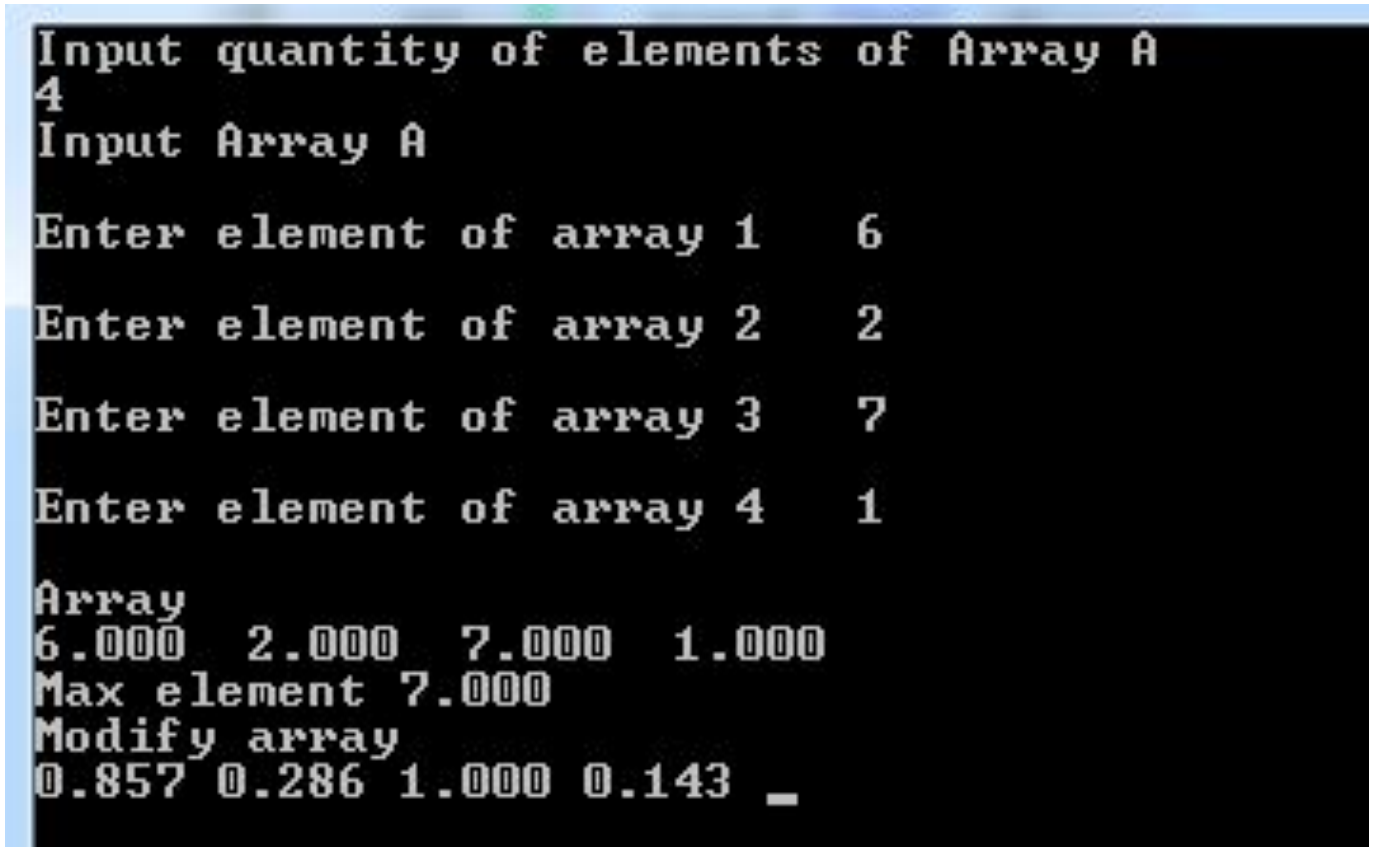
```
printf ("\n\nInput Array B \n");
 InputArray(T,B);
OutArray(T,B);
modifyArray(T,B);
getch();

 printf ("\n\nInput Array C \n");
 InputArray(Y,C);
OutArray(Y,C);
modifyArray(Y,C);
getch();
return 0;
}
```

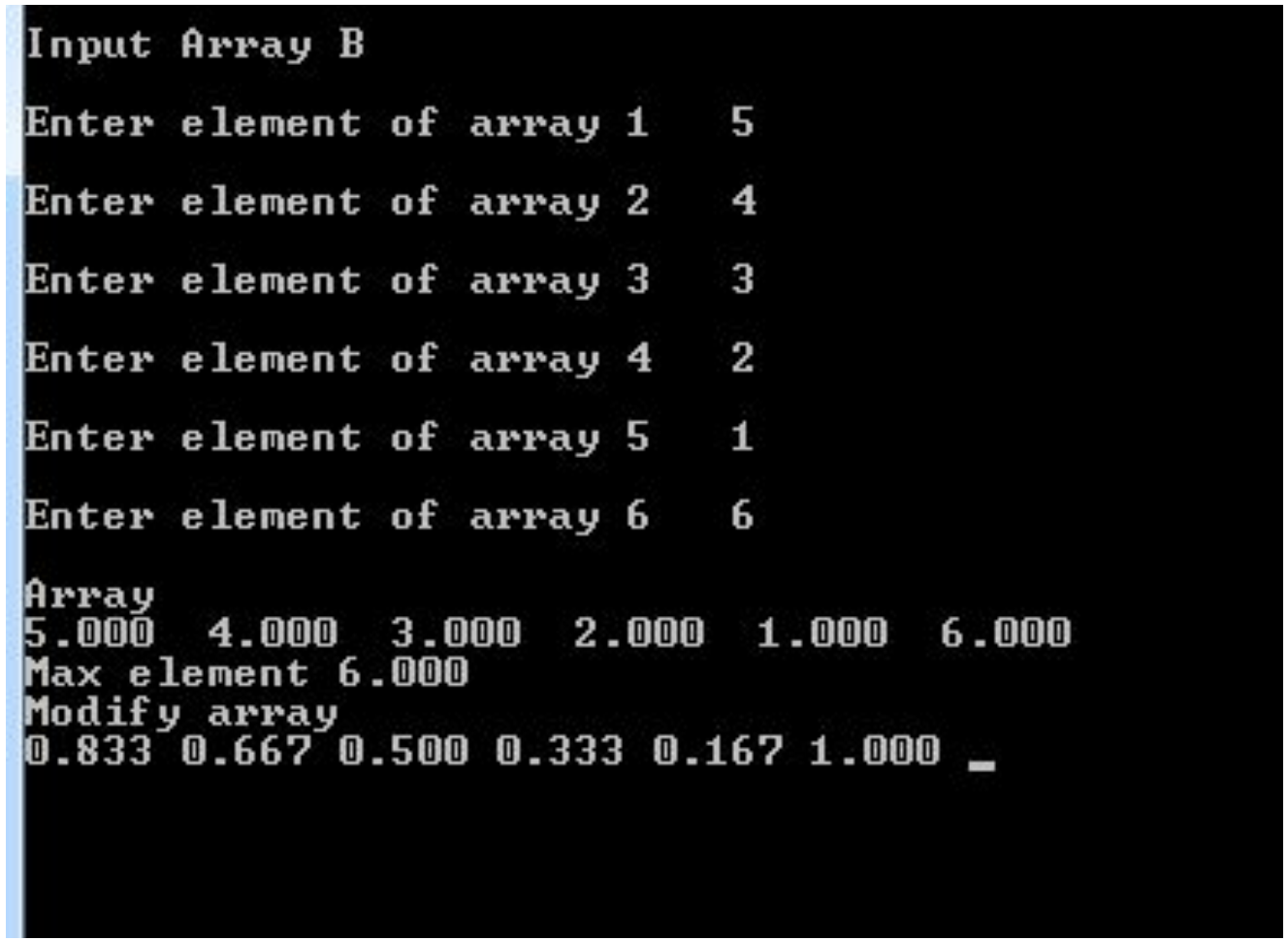
```
void InputArray (int N,float X[N])
{
int i;
for (i=0; i <N; i++)
{
printf("\nEnter element of array %d   ",i+1);
scanf("%f",&X[i]);
}
}
```

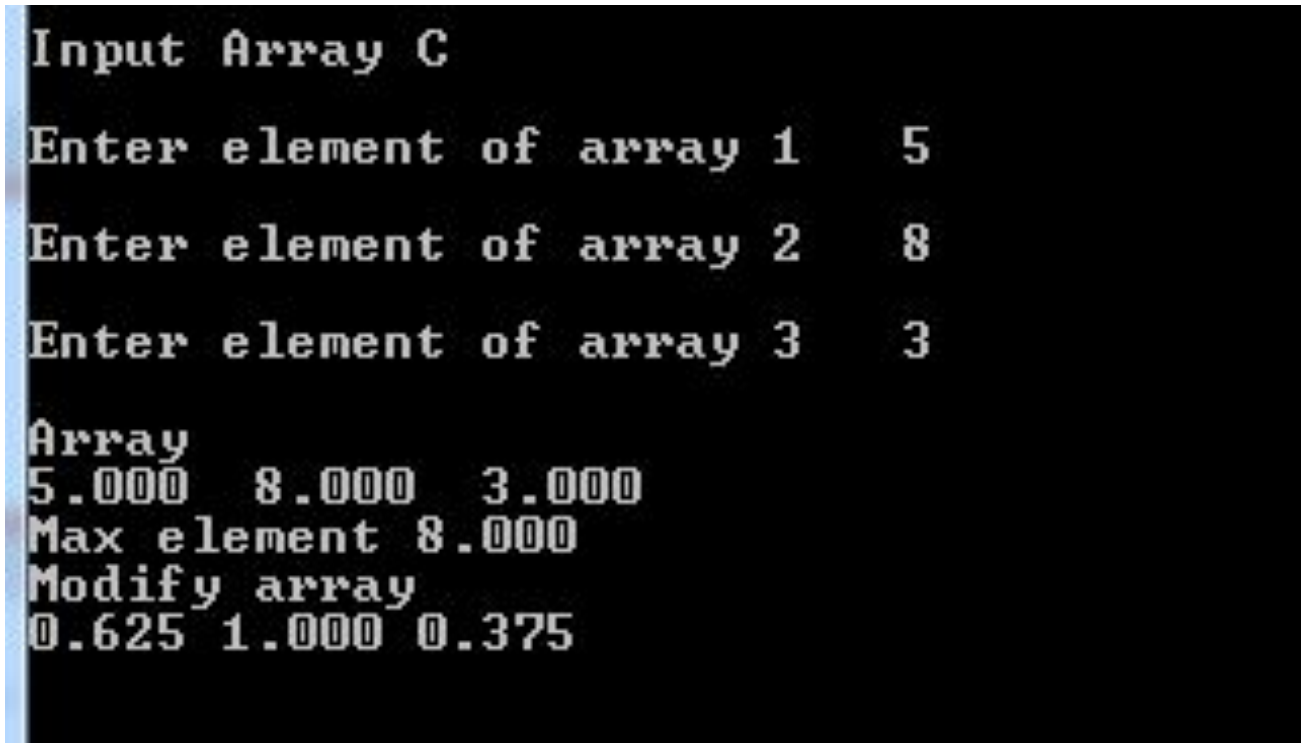
```
void OutArray (int N,float X[N])
 {
 int i;
  printf("\nArray\n" );
  for (i=0; i <N; i++)
  printf("%.3f  ",X[i]);
 }
```

```
void modifyArray(int N, float X[N])
{
    int i;
     float max_array=X[0];
     for(i = 0; i <N; i++)
     {
         if (max_array<X[i]) max_array=X[i];
            }
    printf("\nMax element %.3f ",max_array);
    printf("\nModify array\n");
```

```
 for(i = 0; i <N; i++)
     {
        if (max_array==0) {printf("\nError\n"); break;}
          X[i]=X[i]/max_array;
          printf("%.3f ",X[i]);
           }
}
```


Input quantity of elements of Array A
4
Input Array A
Enter element of array 1 6
Enter element of array 2 2
Enter element of array 3 7
Enter element of array 4 1
Array
6.000 2.000 7.000 1.000
Max element 7.000
Modify array
0.857 0.286 1.000 0.143


Input Array B
Enter element of array 1 5
Enter element of array 2 4
Enter element of array 3 3
Enter element of array 4 2
Enter element of array 5 1
Enter element of array 6 6
Array
5.000 4.000 3.000 2.000 1.000 6.000
Max element 6.000
Modify array
0.833 0.667 0.500 0.333 0.167 1.000

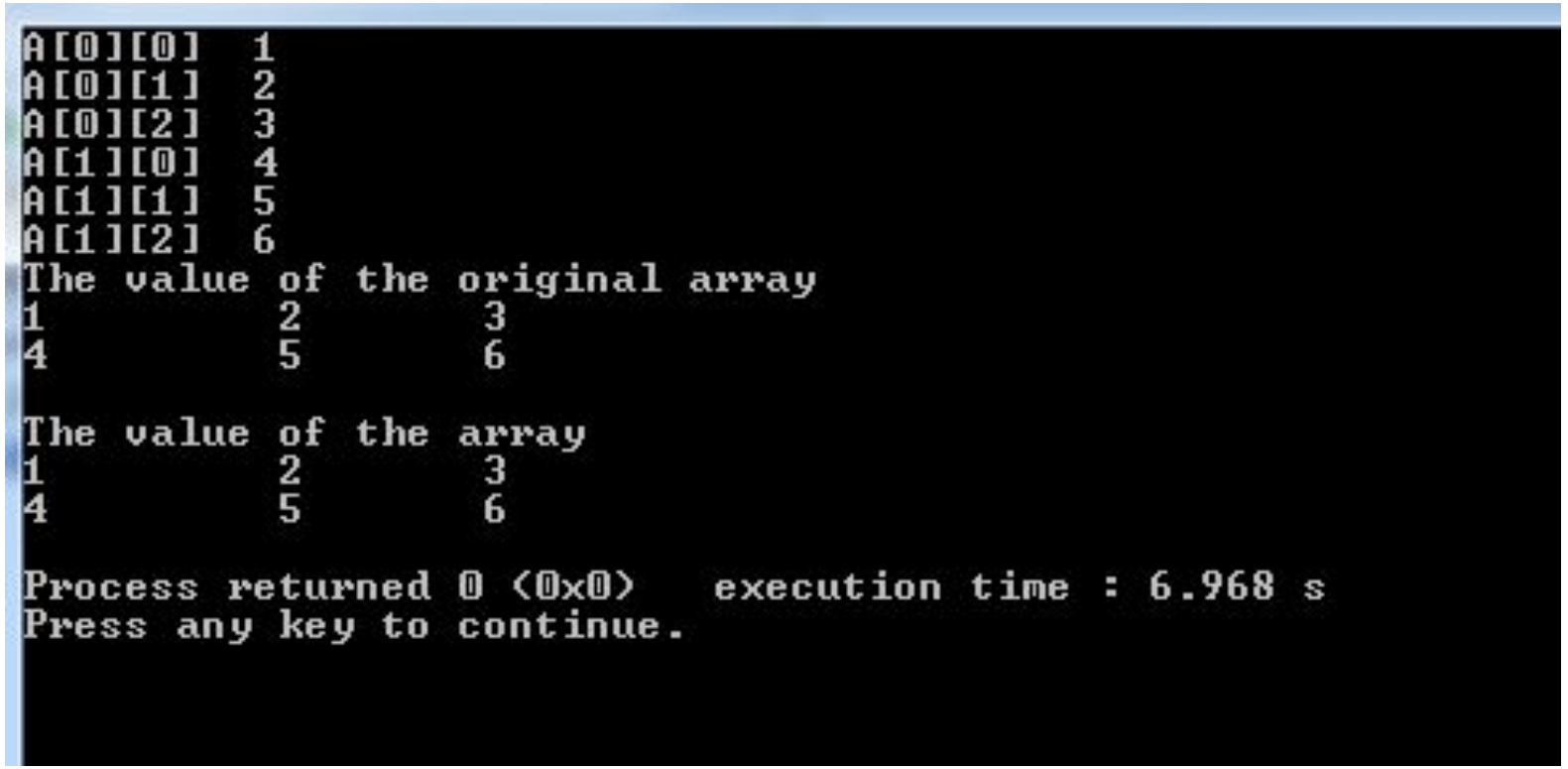

Input Array C
Enter element of array 1 5
Enter element of array 2 8
Enter element of array 3 3
Array
5.000 8.000 3.000
Max element 8.000
Modify array
0.625 1.000 0.375

- Завдання . Ввести двовимірний масив з клавіатури, та вивести масив на екран

```
#include <stdio.h>
#include <stdlib.h>
#define K 2
#define P 3
void input (int AA[K][P])
{
  int i,j;
  for (i = 0; i < K; i++) {
      for (j = 0; j < P; j++) {
      printf("A[%d][%d]   ",i,j);
       scanf("%d",&AA[i][j]);
      }  } }
```

```
void output (int kk,int pp,int AA[kk][pp]){
  int i,j;
  for (i = 0; i < kk; i++) {
      for (j = 0; j < pp; j++) {
       printf("%d\t  ",AA[i][j]);
      }   printf("\n");
  }
  }
```

```
 int main () {


int A[10][10];
input (A);
printf ("The value of the original array\n");
output (K,P,A);
printf("\n");
printf ("The value of the array\n");
output (K,P,A);
return 0;
}
```


A[0][0] 1
A[0][1] 2
A[0][2] 3
A[1][0] 4
A[1][1] 5
A[1][2] 6
The value of the original array
1 2 3
4 5 6
The value of the array
1 2 3
4 5 6
Process returned 0 (0x0) execution time : 6.968 s
Press any key to continue.

```
void output_1 (int kk,int pp,int AA[kk][pp]){
  int i,j;
  for (j = 0; j < pp; j++) {
        printf ("%d column:\n",j+1);
      for (i = 0; i < kk; i++) {
       printf("%d\t  ",AA[i][j]);printf("\n");
      }
  }
  }
```

```
printf ("The value of the array\n");
output_1 (K,P,A);
```