Function in C & Array & rand() & qsort()

1. Transmiterea unor elemente individuale ale unui ARRAY într-o funcție:

```c
#include <stdio.h>
void display(int age1, int age2)
{
    printf("%d\n", age1);
    printf("%d\n", age2);
}

int main()
{
    int ageArray[] = {2, 8, 4, 12};

    // Transmiterea elementului 2 și 3 către funcția display()
    display(ageArray[1], ageArray[2]);
    return 0;
}
```

```c
#include <stdio.h>
void display(int age1, int age2); // declararea functiei

int main()
{
    int ageArray[] = {2, 8, 4, 12};

    // Transmiterea elementului 2 și 3 către funcția display()
    display(ageArray[1], ageArray[2]);
    return 0;
}

void display(int age1, int age2)    // conținutul functiei
{
    printf("%d\n", age1);
    printf("%d\n", age2);
}
```

2. Transmiterea unui vector într-o funcție:

```c
// Programul conține o funcție ce calculează suma elementelor unui vector
// Elementele vectorului sunt citite în main()

#include <stdio.h>
float calculateSum(float number[]);

int main() {
    float result, number[] = {23.4, 55, 22.6, 3, 40.5, 18};

    // Vectorul number este transmis în funcția calculateSum()
    result = calculateSum(number);
    printf("Result = %.2f", result);
    return 0;
}

float calculateSum(float number[]) {

  float sum = 0.0;
  for (int i = 0; i < 6; ++i) {
            sum += number[i];
  }

  return sum;
}
```

Pentru a transmite un vector într-o funcție se va folosi doar numele vectorului în apelul funcției, fără paranteze drepte: `result = calculateSum(number);`, însă în funcția, ce va primi ca parametru un vector, se va indica parantezele drepte după numele vectorului în antetul funcției:

`float calculateSum(float number[]){…}`.

3. Transmiterea unei Matrici într-o funcție:

```c
// transmiterea unui masiv, caz general
#include <stdio.h>
void displayNumbers(int n, int m, int num[n][m]);
int main()
{
    int n, m;
    printf(" n="); scanf("%d", &n);
    printf("\n m="); scanf("%d", &m);
    int num[n][m];
    printf("\n Enter array elements:\n");
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            scanf("%d", &num[i][j]);

    // transmiterea masivului în funcție
    displayNumbers(n, m, num);
    return 0;
}

void displayNumbers(int n, int m, int num[n][m])
{
    printf("Displaying:\n");
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
           printf("%3d", num[i][j]);
        }
        printf("\n");
    }
}
```

4. Generarea numerelor aleatoare

```c
// generarea numerelor aleatoare
// Se va genera numere de la 0 până la INT_MAX
#include <stdio.h>
#include <stdlib.h>

// Driver program
int main(void)
{
    for(int i = 0; i<5; i++)
        printf(" %d ", rand());
    return 0;
}
```

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

int main(void)
{
   int n;
   printf("n = ");
   scanf("%d",&n);
   int randArray[n],i;
   for(i=0;i<n;i++)
     randArray[i]=rand()%100;   //Se generează numere între 0 și 100

   printf("\nElements of the array:\n");
   for(i=0;i<n;i++)
   {
     printf("%d ", randArray[i]);
   }
   return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
int main(void){
   srand(time(0));
   printf("Generarea numerelor aleatoriu: ");
   for(int i = 0; i<5; i++)
      printf(" %d ", rand());
   return 0;
}
```

5. Utilizarea constantelor INT_MAX și INT_MIN. Pentru utilizarea lor este necesar să includem librăria: `#include <limits.h>`.

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

int main(void)
{
    printf("INT_MAX = %d ", INT_MAX);
    printf("\nINT_MIN = %d ", INT_MIN);
    return 0;
}
```

6. Sortarea elementelor unui masiv folosind funcția **qsort()**.

Funcția: void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void*)) sortează un array, unde:

- base – este pointer-ul către primul element al masivului ce trebuie sortat;
- nitems – indică numărul de elemente în masiv;
- size – indică mărimea fiecărui element al masivului;
- compar – numele funcției ce compară două elemente. Această funcție se va scrie de către utilizator în program de fiecare data. Funcția are următorul conținut și nu se va schimba:

```
int compare (const void * a, const void * b) {
   return ( *(int*)a - *(int*)b );
}
```

Funcționalitatea funcției int compare(const void* a, const void* b){...}:

dacă se va returna valoarea:

-1, atunci elementele a și b nu se vor interschimba deoarece a<b;

0, atunci elementele vor rămânea neschimbate, ele sunt egale;

1, atunci elementele a și b se vor interschimba deoarece a>b;

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

int values[] = { 8, 79, 120, 20, 5 };

int compare (const void * a, const void * b) {
   return ( *(int*)a - *(int*)b );
}

int main () {
   int n;

   printf("Masivul până la sortare: \n");
   for( n = 0 ; n < 5; n++ ) {
      printf("%d ", values[n]);
   }

   qsort(values, 5, sizeof(int), compare);

   printf("\nMasivul după sortare: \n");
   for( n = 0 ; n < 5; n++ ) {
      printf("%d ", values[n]);
   }

   return(0);
}
```

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

int values[2][3] = { {88, 2, 100}, {2, 25, 9} };

int compare (const void * a, const void * b) {
   return ( *(int*)a - *(int*)b );
}

int main () {
   int i, j;

   printf("Before sorting the list is: \n");
   for( i = 0 ; i < 2; ++i ) {
       for( j = 0 ; j < 3; ++j ) {
          printf("%d ", values[i][j]);
       }
       printf("\n");
   }
   qsort(values, 6, sizeof(int), compare);

   printf("\nAfter sorting the list is: \n");
   for( i = 0 ; i < 2; ++i ) {
       for( j = 0 ; j < 3; ++j ) {
          printf("%d ", values[i][j]);
       }
       printf("\n");
   }
}
```