Практическое занятие 9. Массивы

Задание 1. Одномерные массивы.

Цель - познакомиться с одномерными массивами базовых типов.

1. В домашнем каталоге создаем и переходим в папку "**TASK_09_01**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_09_01 <Enter>
$cd TASK_09_01 <Enter>
$
```

2. Создаем файл "**array_1.c**" и добавляем в него следующие строки:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  int a[5] = {1, 2, 3, 4, 5};
  int la, le, i;

  // Вычислим размер элемента массива
  le = sizeof(a[0]);
  printf("element size = %d\n", le);

  // Длина массива может быть вычислена функцией sizeof()
  la = sizeof(a) / le;
  printf("Array length = %d\n", la);

  // Выведем на экран все элементы массива
  for(i = 0; i < la; i++){
        printf("a[%d]= %d\n", i, a[i]);
  }

  return 0;
}
```

3. Самостоятельно собрать/запустить программу.

4. Самостоятельно вычислить среднее значение и дисперсию элементов массива "**a**".

Задание 2. Массивы. Сортировка.

Цель - отсортировать массив методом *пузырька*.

1. В домашнем каталоге создаем и переходим в папку "**TASK_09_02**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_09_02 <Enter>
$cd TASK_09_02 <Enter>
$
```

2. Создаем файл "**bubble.c**" и добавляем в него следующие строки:

```
#include <stdio.h>

#define FALSE 0
#define TRUE  (!FALSE)

int main(int argc, char *argv[]) {
    float a[] = {9.0f, 2.0f, 3.0f, 4.0f, 5.0f,
                 6.0f, 7.0f, 8.0f, 1.0f, 0.0f};
    float tmp;
    unsigned int i, j;
    char flag;
    int len = sizeof(a) / sizeof(float);

    // Выводи массив
    for (i = 0; i < len; i++) {
        printf("%.3f\n", a[i]);
    }
    printf("\n");

    // Пока массив не отсортирован
    do {
        flag = FALSE;
        // Проходим по массиву. Если следующий элемент больше
        // предыдущего, то меняем их местами и по новой проверяем
        // массив
        for (i = 1; i < len; i++) {
            if (a[i] > a[i - 1]) {
                tmp = a[i];
                a[i] = a[i - 1];
                a[i - 1] = tmp;
                flag = TRUE;
            }
        }

    } while(flag == TRUE);

    // Выводим отсортированный массив
    for (i = 0; i <len; i++) {
        printf("%.3f\n", a[i]);
    }

    return 0;
}
```

3. Самостоятельно разобраться/собрать/запустить программу.

Задание 3. Многомерные массивы.

Цель - познакомиться с многомерными массивами.

1. В домашнем каталоге создаем и переходим в папку "**TASK_09_03**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_09_03 <Enter>
$cd TASK_09_03 <Enter>
$
```

2. Создаем файл "**array_2.c**" и добавляем в него следующие строки:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    // Создадим и проанализируем трехмерный массив
    int A[2][3][4] = {
      {
          {111, 112, 113, 114},
          {121, 122, 123, 124},
          {131, 132, 133, 134}
      },
      {
          {211, 212, 213, 214},
          {221, 222, 223, 224},
          {231, 232, 233, 234}
      }
    };

    // Определим размер и вычислим размеренности массива
    int size = sizeof(A);
    int d1 = sizeof(A) / sizeof(A[0]);
    int d2 = sizeof(A[0]) / sizeof(A[0][0]);
    int d3 = sizeof(A[0][0]) / sizeof(A[0][0][0]);

    printf("size = %d, d1 = %d, d2 = %d, d3 = %d\n", size, d1, d2, d3);

    return 0;
}
```

3. Самостоятельно разобраться/собрать/запустить программу.

4. Самостоятельно организовать вывод на экран содержимого массива трехмерного массива "**A**".

Задание 4. Многомерные массивы.

Цель - составить программу умножения матриц.

1. В домашнем каталоге создаем и переходим в папку "**TASK_09_04**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_09_04 <Enter>
$cd TASK_09_04 <Enter>
$
```

2. Создаем файл "**m1xm2.c**" и добавляем в него следующие строки:

```
#include <stdio.h>

```

```
int main(int argc, char *argv[]) {
  // Первая матрица размером 2x3
  int m1[2][3] = {
        {1, 1, 1},
        {1, 1, 1}
  };

  // Вторая матрица размером 3x2
  int m2[3][2] = {
        {1, 1},
        {1, 1},
        {1, 1}
  };

  // Третья матрица
  int m3[2][2] = {0};

  // N0 - количество строк в первой матрице и столбцов во второй
  // Произведение матриц будет иметь размер N0xN0
  int N0 = sizeof(m1) / sizeof(m1[0]);
  // N1 - количество столбцов в первой матрице и строк во второй
  int N1 = sizeof(m1[0]) / sizeof(m1[0][0]);
  int i, j, c, r;

  // Умножение матриц: "строка на столбец"
  for(r = 0; r < N0; r++){
     for(c = 0; c < N0; c++){
         for(i = 0; i < N1; i++){
               m3[r][c] = m3[r][c] + m1[r][i] * m2[i][c];
        }
     }
   }

   // Выводим на экран элементы матрицы
   for(r = 0; r < N0; r++){
      for(c = 0; c < N0; c++){
          printf("%d, ", m3[r][c]);
      }
      printf("\n");
   }

   return 0;
}
```

3. Самостоятельно разобраться/собрать/запустить программу.

4. Самостоятельно модифицировать/собрать/запустить программу для умножения матриц размера 3x4 и 4x3.