Лабораторная работа 3.1
Указатели и Статические массивы

Задание:
Изучить теоретический материал по темам:

1. Виды массивов
2. Статические массивы
3. Алгоритм сортировки пузырьком
4. Алгоритм сортировки подсчетом
5. Алгоритм сортировки слиянием

Написать программу с меню, которая по запросу пользователя выполняет одно из следующих действий

1. Сортирует числовой массив (длины не более 1000) по возрастанию по алгоритму пузырьковой сортировки. Саму сортировку написать отдельной функцией. Массив, введенный пользователем передавать в функцию сортировки.
2. Сортирует символьный массив (буквы a-z) (длины не более 1000) по возрастанию по алгоритму сортировки подсчетом. Саму сортировку написать отдельной функцией. Массив, введенный пользователем передавать в функцию сортировки.
3. Сортирует числовой массив (длины не более 1000) по возрастанию по алгоритму сортировки слиянием. Саму сортировку написать отдельной функцией. Массив, введенный пользователем передавать в функцию сортировки.

Критерии оценивания: сделано меню и 1 из сортировок – удовлетворительно, сделано меню и 2 сортировки – хорошо, все выполнено – отлично. Если сортировки не реализованы в виде отдельных функций и/или массив вводится внутри самой функции сортировки, то оценка снижается на один балл.

Форма представления выполненных заданий: файл *Ivanov_Ivan_111_111_lab31.cpp* содержащий решение поставленных задач.

Теоретические сведения:

Указатели

Переменная, хранящая адрес другой переменной в памяти, называется указателем (pointer).
Синтаксис объявления указателей

```
int *ptr_int = &a;
double * ptr_dbl = &dbl;
```

Оператор амперсенд & - оператор получения адреса.

Все указатели всегда одной и той же разрядности (размера). Каким бы ни был огромным тип данных (строка, объект, массив) указатель на него всегда будет занимать 4/8 байт, что и является основным преимуществом указателей. Для проверки можно запустить код:

```
std::cout << "int pointer size = " << sizeof(ptr_int) << std::endl;
std::cout << "double pointer size = " << sizeof(ptr_dbl) << std::endl;
std::cout << "char pointer size = " << sizeof(char*) << std::endl;
std::cout << "long long pointer size = " << sizeof(long long *) << std::endl;
std::cout << "bool pointer size = " << sizeof(bool *) << std::endl;
```

Содержимое указателей

```
std::cout << std::endl << "ptr_int = 0x" << std::hex << ptr_int << std::endl;
std::cout << "ptr_ch = 0x" << ptr_dbl << std::endl;
```

std::hex включает отображение целых чисел по 16-ричной системе пока не будет включена другая система счисления, подобным образом все числа будут печататься в x16 системе.

В конечном итоге компилятор, преобразуя текст программы, заменяет имена переменных на их смещения (адреса), т.к. процессор обращается к данным именно по адресу.
Оператор разыменования указателей: чтобы получить значение, хранящееся по данному указателю, нужно ещё раз применить оператор *

```
std::cout << std::endl << "*ptr_int = " << std::dec << *ptr_int << std::endl;
std::cout << "* ptr_dbl = " << *ptr_dbl << std::endl;
```

Массивы

Массив — это область памяти, где могут последовательно храниться несколько значений. Массив — блок из нескольких однотипных данных.
Задавать массивом удобно:

1) вектора и матрицы
2) множество точек из 3D и 2D геометрии
3) просто какой-либо числовой ряд, таблица и т.д.
4) строка - массив символов
5) видеобуфер (двумерный массив, соответствующий пикселам экрана)

Статические массивы – массивы фиксированной длины. Длина динамических массивов может изменяться на протяжении программы.

Объявление массива:

```
int arr[5];// 5 – длина
int arr1[5] = {0};//все элементы массива равны 0
int arr2[5] = { 1, 50, 11, 12, 101}; // массив задан поэлементно
```

В оперативной памяти массив хранится компактно (без разрывов, элемент за элементом), нумеруются элементы массива с 0.

Вывод массива:

```
std::cout << arr[0] << '\t' <<
   arr[1] << '\t' << //в одинарных кавычках - только 1 символ (char), в двойных - любое множество
   arr[2] << '\t' << // в общем случае можно везде использовать двойные
   arr[3] << '\t' <<
   arr[4] << std::endl;

std::cout << "arr = " << arr << std::endl;
std::cout << "*arr = arr[0] = " << *arr << std::endl;
std::cout << "*(arr+4) = arr[4] = " << *(arr+4) << std::endl;
//!!! выход за границы допустимой памяти i +- 10000
```

Двумерный массив:

Статические массивы могут быть двумерными:

```
int arr2D[3][3] = { // инициализируются построчно
                    {1, 2, 3},
                    {4, 5, 6},
                    {7, 8, 9}
                  };
```

```
        int arr2D_cin [2][3]={0};
//Инициализация массива пользователем
        for (int i = 0; i < 2; i++)
        {
                for (int j = 0; j < 3; j++)
                {
                        std::cin >> arr2D_cin[i][j];
                }
        }
```

Если бы захотели занулить весь массив, написали бы int arr2D[3][3] = {0}. Размерности массивов задаются статически, нельзя задать размер массива с помощью переменной int arr2D[a][3]; для этого нужно воспользоваться динамической памятью и совсем другим синтаксисом объявления.

Вывод массива на печать

```
        for (int i = 0; i<3; i++)
        {
                for (int j = 0; j < 3; j++)
                {
                        std::cout << arr2D[i][j] << '\t';
                }
                std::cout << std::endl;
        }
```

По сути, квадратные скобки работают как разыменование и смещение:

```
        std::cout << "*(*(arr2D+2)+2) = arr2D[2][2] = " << *(*(arr2D + 2) + 2) << std::endl;
```

У arr2D тип **int (указатель на указатель на int), arr2D[1] тип *int - указатель на одну строчку - одномерный массив внутри двумерного. Двумерный массив уложен в одномерном пространстве оперативной памяти построчно.

Алгоритм пузырьковой сортировки:

```
Функция СортировкаПузырьком(Знач Массив)

    Для i = 0 По Массив.ВерхГраница() Цикл
        Для j = 0 ПО Массив.Верхграница() - i - 1 Цикл
            Если Массив[j] > Массив[j + 1] Тогда
                Замена = Массив[j];
                Массив[j] = Массив[j + 1];
                Массив[j + 1] = Замена;
            КонецЕсли;
        КонецЦикла;
    КонецЦикла;
    Возврат Массив;
КонецФункции
```

Алгоритм сортировки подсчетом:

```
Функция СортировкаПодсчетом(Знач Массив)
    КоличествоРазлЭлем = 26
    Создать целочисленный массив Подсчет[КоличествоРазлЭлем]
    Для i = 0 По Массив.ВерхГраница() Цикл
```

```
        j=Целое(Массив[i]-‘a’) //порядковый номер очередной буквы
        Подсчет[j]++;
    КонецЦикла;
    i=0
    Пока j <= КоличествоРазлЭлем Цикл
        Если Подсчет[j]>0
            Массив[i]=Символ(Целое(‘a’)+j) //порядковый номер очередной буквы
            i++;
            Подсчет[j]--;
        Иначе
            j++;
        КонецЕсли;
    КонецЦикла;
    Возврат Массив;
КонецФункции
```

Сортировка слиянием:

1. массив рекурсивно разбивается пополам, и каждая из половин делиться до тех пор, пока размер очередного подмассива не станет равным единице;
2. далее выполняется операция алгоритма, называемая слиянием. Два единичных массива сливаются в общий результирующий массив, при этом из каждого выбирается меньший элемент (сортировка по возрастанию) и записывается в свободную левую ячейку результирующего массива. После чего из двух результирующих массивов собирается третий общий отсортированный массив, и так далее. В случае если один из массивов закончиться, элементы другого дописываются в собираемый массив;
3. в конце операции слияния, элементы перезаписываются из результирующего массива в исходный.