

## ТЕМА: ВВЕДЕНИЕ В ЯЗЫК ПРОГРАММИРОВАНИЯ «C++»

### Домашнее задание 2

#### ЗАДАНИЕ 1

Написать программу «справочник». Создать два одномерных массива. Один массив хранит номера мобильных телефонов, второй — домашние телефонные номера.

Реализовать меню для пользователя:

- Отсортировать по номерам мобильных
- Отсортировать по домашним номерам телефонов;
- Вывести пользовательский данные;
- Выход.

##### Подсказка 1

Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение»

##### Решение 1

#### ЗАДАНИЕ 2

Написать программу, реализующую сортировку массива с помощью усовершенствованной сортировки пузырьковым методом. Усовершенствование состоит в том, чтобы анализировать количество перестановок на каждом шагу, если это количество равно нулю, то продолжать сортировку нет смысла — массив отсортирован.

##### Подсказка 2

Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение»

##### Решение 2

### ЗАДАНИЕ 3

Есть стопка оладий различного радиуса. Единственная операция, проводимая с ними — между любыми двумя сеем лопатку и меняем порядок оладий над лопаткой на обратный. Необходимо за минимальное количество операций таких отсортировать снизу вверх по убыванию радиуса.

### ЗАДАНИЕ 4

Написать программу, которая сравнивает число перестановок элементов при использовании сортировки «пузырьком» (усовершенствованная версия задания 2) и методом выбора. Выполнить ее тестирование на разных 10 массивах, содержащих 1000 случайных элементов, вычислить среднее число перестановок для каждого метода после тестирования.

Финальные задания рассчитаны на самостоятельное решение. Здесь не будет подсказок и готового алгоритма. Вам необходимо применить все практические навыки, полученные из предыдущих заданий.

**ПОДСКАЗКА К ЗАДАНИЮ 1**

1. При объявлении массива необходимо указать типа и его размерность. Если оба массива хранят номера одних и тех же пользователей и количество пользователей, то как связаны размерности массивов и количество пользователей?
2. Номера телефонов (значения элементов массивов) можно представлять в виде целых чисел.
3. Операции сортировки и вывода данных возможны только после заполнения обоих массивов.
4. Как длина внешнего цикла (число повторений сортировки) в алгоритме сортировки пузырьком зависит от длины массива, который нужно отсортировать?
5. Как длина внутреннего цикла («всплытие пузырька») зависит от переменной-счётчика внешнего цикла?
6. Если сортируются полностью все данные о пользователях, то переставляются (при необходимости) не только элементы массива, который является ядром сортировки, но и соответствующие элементы другого массива.

**ПОДСКАЗКА К ЗАДАНИЮ 2**

1. При объявлении массива необходимо указать его тип и размерность.
2. Как длина внешнего цикла (число повторений сортировки) в алгоритме сортировки пузырьком зависит от длины массива, который нужно отсортировать?
3. Как длина внутреннего цикла («всплытие пузырька») зависит от переменной-счётчика внешнего цикла?
4. Анализ количества перестановок на каждом шагу — это проверка количества перестановок после очередного прохода алгоритма (шага внешнего цикла).
5. В начале каждого прохода переменную — счетчик количества перестановок нужно обнулять.

## РЕШЕНИЕ ЗАДАНИЯ 1

1. Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем необходимое число переменных и констант.

Так как длина обоих массивов — это количество пользователей, о которых есть информация в справочнике (т.к. по каждому пользователю в справочнике должна быть информация и о номере мобильного телефона, и о номере домашнего телефона), то достаточно одной целочисленной константы для хранения длины обоих массивов.

В нашем примере длина массивов будет 5, т. е. количество пользователей, о которых есть информация в справочнике — пять.

Далее объявляем два целочисленных массива, используя созданную ранее целочисленную константу для задания размеров массивов.

Также (т. к. предполагается реализация меню) нам понадобится переменная для хранения номера пункта меню, который выбрал пользователь.

Учитывая, что операции сортировки и вывода данных возможны только после заполнения обоих массивов, нам необходима переменная — признак заполнения справочника данными (например, `false` — данные еще не введены в справочник, `true` — данные введены).

Далее, в случае выбора пользователем операций сортировки или вывода данных, будем проверять значение этой переменной признака и, если данные еще не были введены, то уведомлять пользователя об этом и останавливать операцию вывода (сортировки).

Реализуем цикл для повторения вывода меню. Это будет цикл с постусловием, так как вначале необходимо хотя бы раз отобразить меню, а потом проверять, следует ли его повторять. Повторение вывода меню происходит, если пользователь ввел непредусмотренный элемент меню или после выполнения одного из пунктов меню. Для остановки программы следует предусмотреть пункт «Выход».

Меню будет содержать пять пунктов: ввод данных, вывод данных, сортировка по номерам мобильных телефонов, сортировка по номерам домашних телефонов и пятый пункт меню — выход.

В каждом кейсе, кроме первого (ввод данных) и последнего (выход) будем проверять состояние переменной признака заполненных данных.

Для улучшения читабельности кода реализуем пункты меню с использованием перечисляемого типа данных. Каждому перечислителю автоматически присваивается целочисленное значение в зависимости от его позиции в списке перечисления. По умолчанию, первому перечислителю присваивается целое число 0, а каждому следующему — на единицу больше, чем предыдущему. Так как нам необходимо задать для первого элемента меню «Ввод данных» значение 1, то выполним присваивание нужных нам значений перечислителей.

```
#include <iostream>

using namespace std;

enum menuItems
{
```

```
ENTER_DATA = 1,
OUTPUT_DATA=2,
SORT_BY_MOBILE_NUM=3,
SORT_BY_TEL_NUM = 4,
QUIT=5
};

int main()
{
    cout << "Home task #9.2.1\n";

    const int n = 5;
    int userChoice, mobTel[n], tel[n], temp;
    bool isDataEntered = false;

    do {
        cout << "Your choice:\n";
        cout << "1 - enter data\n";
        cout << "2 - output data\n";
        cout << "3 - sort by mobile number\n";
        cout << "4 - sort by tel.number\n";
        cout << "5 - quit\n";
        cin >> userChoice;

        switch (userChoice) {
            case ENTER_DATA:
            {
                cout << "Please, enter data for each "
                    << "customer\n";
                isDataEntered = true;
                break;
            }
            case OUTPUT_DATA:
            {
```

```

        if (isDataEntered)
        {
            cout << "Customer's data:\n";
        }
        else
        {
            cout << "There is no data!" << "\n";
            cout << "Please, enter data for each "
                "customer at first\n";
        }

        break;
    }
    case SORT_BY_MOBILE_NUM:
    {
        if (isDataEntered)
        {
            cout << "Sorting data by mobile "
                "number ....\n";
        }
        else
        {
            cout << "There is no data!" << "\n";
            cout << "Please, enter data for each "
                "customer at first\n";
        }

        break;
    }
    case SORT_BY_TEL_NUM:
    {
        if (isDataEntered)
        {

```



```
        cout << "Sorting data by tek. "
            "number ....\n";
    }
    else
    {
        cout << "There is no data!" << "\n";
        cout << "Please, enter data for each "
            "customer at first\n";
    }

    break;
}

case QUIT:
{
    cout << "See you!";
    break;
}
default:
    cout << "Wrong menu item!";
}
} while (userChoice != 5);

return 0;
}
```

2. Если пользователь выбрал первый пункт меню (ввод данных), организуем цикл (длина которого равна количеству пользователей), в котором для каждого пользователя попросим ввести его мобильный и домашний номера телефонов, занесем введенную информацию в соответствующие массивы.

```
#include <iostream>

using namespace std;

enum menuItems
{
    ENTER_DATA = 1,
    OUTPUT_DATA=2,
    SORT_BY_MOBILE_NUM=3,
    SORT_BY_TEL_NUM = 4,
    QUIT=5
};

int main()
{
    cout << "Home task #9.2.1\n";

    const int n = 5;
    int userChoice, mobTel[n], tel[n], temp;
    bool isDataEntered = false;

    do {
        cout << "Your choice:\n";
        cout << "1 - enter data\n";
        cout << "2 - output data\n";
        cout << "3 - sort by mobile number\n";
        cout << "4 - sort by tel.number\n";
        cout << "5 - quit\n";
        cin >> userChoice;

        switch (userChoice) {
            case ENTER_DATA:
            {
                cout << "Please, enter data for each "
                    "customer\n";
```

```
isDataEntered = true;
for (int i = 0; i < n; i++)
{
    cout << "Input mobile number for ";
    cout << i + 1 << " customer:\n";
    cin >> mobTel[i];
    cout << "Input tel. number for ";
    cout << i + 1 << " customer:\n";
    cin >> tel[i];
}

break;
}

case OUTPUT_DATA:
{
    if (isDataEntered)
    {
        cout << "Customer's data:\n";
    }
    else
    {
        cout << "There is no data!" << "\n";
        cout << "Please, enter data for each "
            "customer at first\n";
    }

    break;
}

case SORT_BY_MOBILE_NUM:
{
    if (isDataEntered)
    {
```

```
        cout << "Sorting data by mobile "
              "number ....\n";
    }
    else
    {
        cout << "There is no data!" << "\n";
        cout << "Please, enter data for "
              "each customer at first\n";
    }
    break;
}
case SORT_BY_TEL_NUM:
{
    if (isDataEntered)
    {
        cout << "Sorting data by tek. "
              "number ....\n";
    }
    else
    {
        cout << "There is no data!" << "\n";
        cout << "Please, enter data for "
              "each customer at first\n";
    }
    break;
}
case QUIT:
{
    cout << "See you!";
    break;
}
default:
    cout << "Wrong menu item!";
}
```

```
    } while (userChoice != 5);  
    return 0;  
  
}
```

3. Для реализации сортировки будем использовать алгоритм «пузырька», т. е. будем анализировать массив от начала до конца, меняя местами неотсортированные соседние элементы. В результате первого прохода подобным образом на последнее место в массиве «всплывёт» минимальный элемент (т.к. сортировка производится по убыванию). Далее повторяем обход неотсортированной части массива (от первого элемента до предпоследнего, т.к. последний уже на своем месте) и меняем по пути неотсортированных соседей. Количество проходов равно количеству элементов массива за вычетом единицы, т.к. происходит попарное сравнение. Для реализации таких проходов массива организуем внешний цикл, внутри которого располагаем цикл для попарного сравнения элементов. Длина внутреннего цикла будет на каждом проходе уменьшаться на единицу, т.к. за один проход один «пузырек» всплывает на свое место и число неотсортированных элементов сокращается. Так как сортируются полностью все данные о пользователях, то переставляются (при необходимости) не только элементы массива, который является ядром сортировки, но и соответствующие элементы другого массива. Таким образом, сравнение производится на одном массиве, а перестановка (при необходимости) в обоих.

Добавляем во второй и третий кейсы реализации алгоритма сортировки «пузырьком»: во втором кейсе анализируется массив с номерами мобильных телефонов, а в третьем — с номерами домашних телефонов.

```
#include <iostream>

using namespace std;
enum menuItems
{
    ENTER_DATA = 1,
    OUTPUT_DATA=2,
    SORT_BY_MOBILE_NUM=3,
    SORT_BY_TEL_NUM = 4,
    QUIT=5
};

int main()
{
    cout << "Home task #9.2.1\n";
    const int n = 5;
    int userChoice, mobTel[n], tel[n], temp;
    bool isDataEntered = false;

    do {
        cout << "Your choice:\n";
        cout << "1 - enter data\n";
        cout << "2 - output data\n";
        cout << "3 - sort by mobile number\n";
        cout << "4 - sort by tel.number\n";
        cout << "5 - quit\n";
        cin >> userChoice;

        switch (userChoice) {
```

```
case ENTER_DATA:
{
    cout << "Please, enter data for each "
           "customer\n";
    isDataEntered = true;

    for (int i = 0; i < n; i++)
    {
        cout << "Input mobile number for ";
        cout << i + 1 << " customer:\n";
        cin >> mobTel[i];
        cout << "Input tel. number for ";
        cout << i + 1 << " customer:\n";
        cin >> tel[i];
    }

    break;
}

case OUTPUT_DATA:
{
    if (isDataEntered)
    {
        cout << "Customer's data:\n";
    }
    else
    {
        cout << "There is no data!" << "\n";
        cout << "Please, enter data for each "
               "customer at first\n";
    }

    break;
}
```

```
case SORT_BY_MOBILE_NUM:
{
    if (isDataEntered)
    {
        cout << "Sorting data by mobile "
              "number ....\n";
        for (int i = 1; i < n; ++i)
        {
            for (int r = 0; r < n - i; r++)
            {
                if (mobTel[r] < mobTel[r + 1])
                {
                    temp = mobTel[r];
                    mobTel[r] = mobTel[r + 1];
                    mobTel[r + 1] = temp;
                    temp = tel[r];
                    tel[r] = tel[r + 1];
                    tel[r + 1] = temp;
                }
            }
        }
    }
    else
    {
        cout << "There is no data!" << "\n";
        cout << "Please, enter data
              "for each customer at first\n";
    }
    break;
}

case SORT_BY_TEL_NUM:
{
    if (isDataEntered)
```



```
{
    cout << "Sorting data by tek. "
           "number ....\n";
    for (int i = 1; i < n; ++i)
    {
        for (int r = 0; r < n - i; r++)
        {
            if (tel[r] < tel[r + 1])
            {
                temp = tel[r];
                tel[r] = tel[r + 1];
                tel[r + 1] = temp;

                temp = mobTel[r];
                mobTel[r] = mobTel[r + 1];
                mobTel[r + 1] = temp;
            }
        }
    }
}
else
{
    cout << "There is no data!" << "\n";
    cout << "Please, enter data for "
           "each customer at first\n";
}
break;
}
case QUIT:
{
    cout << "See you!";
    break;
}
```

```
        default:
            cout << "Wrong menu item!";
        }
    } while (userChoice != 5);

    return 0;
}
```

4. Реализуем пункт меню «Вывод данных». Т.к. длины массивов совпадают, то организуем только один цикл, на каждом шаге которого выводим полную информацию об i-том пользователе.

```
#include <iostream>

using namespace std;

enum menuItems
{
    ENTER_DATA = 1,
    OUTPUT_DATA=2,
    SORT_BY_MOBILE_NUM=3,
    SORT_BY_TEL_NUM = 4,
    QUIT=5
};

int main()
{
    cout << "Home task #9.2.1\n";

    const int n = 5;
    int userChoice, mobTel[n], tel[n], temp;
```

```
bool isDataEntered = false;
do {
    cout << "Your choice:\n";
    cout << "1 - enter data\n";
    cout << "2 - output data\n";
    cout << "3 - sort by mobile number\n";
    cout << "4 - sort by tel.number\n";
    cout << "5 - quit\n";
    cin >> userChoice;

    switch (userChoice) {
        case ENTER_DATA:
        {
            cout << "Please, enter data for each "
                  << "customer\n";
            isDataEntered = true;

            for (int i = 0; i < n; i++)
            {
                cout << "Input mobile number for ";
                cout << i + 1 << " customer:\n";
                cin >> mobTel[i];
                cout << "Input tel. number for ";
                cout << i + 1 << " customer:\n";
                cin >> tel[i];
            }

            break;
        }

        case OUTPUT_DATA:
        {
            if (isDataEntered)
            {
```

```

        cout << "Customer's data:\n";
        cout << "Cust.ID\tMobile number\tTel."
              "number\n";

        for (int i = 0; i < n; i++)
        {
            cout << i + 1<<" " << mobTel[i];
            cout << " " << tel[i] << "\n";
        }
    }
    else
    {
        cout << "There is no data!" << "\n";
        cout << "Please, enter data for each "
              "customer at first\n";
    }

    break;
}

case SORT_BY_MOBILE_NUM:
{
    if (isDataEntered)
    {
        cout<< "Sorting data by mobile "
              "number ....\n";
        for (int i = 1; i < n; ++i)
        {
            for (int r = 0; r < n - i; r++)
            {
                if (mobTel[r] < mobTel[r + 1])
                {
                    temp = mobTel[r];
                    mobTel[r] = mobTel[r + 1];
                    mobTel[r + 1] = temp;
                }
            }
        }
    }
}

```

```

        temp = tel[r];
        tel[r] = tel[r + 1];
        tel[r + 1] = temp;
    }
}
}
else
{
    cout << "There is no data!" << "\n";
    cout << "Please, enter data for "
           "each customer at first\n";
}
break;
}
case SORT_BY_TEL_NUM:
{
    if (isDataEntered)
    {
        cout << "Sorting data by tek. "
               "number ....\n";

        for (int i = 1; i < n; ++i)
        {
            for (int r = 0; r < n - i; r++)
            {
                if (tel[r] < tel[r + 1])
                {
                    temp = tel[r];
                    tel[r] = tel[r + 1];
                    tel[r + 1] = temp;

                    temp = mobTel[r];

```

```
        mobTel[r] = mobTel[r + 1];
        mobTel[r + 1] = temp;
    }
}
}
else
{
    cout << "There is no data!" << "\n";
    cout << "Please, enter data for each "
           "customer at first\n";
}
break;
}
case QUIT:
{
    cout << "See you!";
    break;
}
default:
    cout << "Wrong menu item!";
}
} while (userChoice != 5);

return 0;

}
```

Результаты работы программы (в консоли):

- Тест 1 — Пользователь выбрал несуществующий пункт меню

```
Home task #9.2.1
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
7
Wrong menu item!Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
_
```

Рисунок 1

- Тест 2 — Пользователь выбрал пункт «Вывод данных» до их ввода

```
Home task #9.2.1
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
2
There is no data!
Please, enter data for each customer at first
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
```

Рисунок 2

- Тест 3 — Пользователь выбрал пункт «Сортировка по номерам мобильных телефонов» до ввода данных

```
Home task #9.2.1
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
2
There is no data!
Please, enter data for each customer at first
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
3
There is no data!
Please, enter data for each customer at first
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
```

Рисунок 3



- Тест 4 — Пользователь выбрал пункт «Ввод данных» и затем пункт «Вывод данных»

```
Home task #9.2.1
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
1
Please, enter data for each customer
Input mobile number for 1 customer:
0956789450
Input tel. number for 1 customer:
3456789
Input mobile number for 2 customer:
0672345678
Input tel. number for 2 customer:
2335689
Input mobile number for 3 customer:
0973456790
Input tel. number for 3 customer:
3314567
Input mobile number for 4 customer:
0634567891
Input tel. number for 4 customer:
3451212
Input mobile number for 5 customer:
0974567890
Input tel. number for 5 customer:
5567890
```

Рисунок 4

```

Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
2
Customer's data:
Cust.ID Mobile number   Tel. number
1 956789450 3456789
2 672345678 2335689
3 973456790 3314567
4 634567891 3451212
5 974567890 5567890
    
```

Рисунок 5

- Тест 5 — Пользователь выбрал пункт «Сортировка по номерам мобильных телефонов» и затем пункт «Вывод данных»

```

Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
3
Sorting data by mobile number ....
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
2
Customer's data:
Cust.ID Mobile number   Tel. number
1 974567890 5567890
2 973456790 3314567
3 956789450 3456789
4 672345678 2335689
5 634567891 3451212
    
```

Рисунок 6

- Тест 6 — Пользователь выбрал пункт «Сортировка по номерам домашних телефонов» и затем пункт «Вывод данных»

```
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
4
Sorting data by tek. number ....
Your choice:
1 - enter data
2 - output data
3 - sort by mobile number
4 - sort by tel.number
5 - quit
2
Customer's data:
Cust.ID Mobile number  Tel. number
1 974567890 5567890
2 956789450 3456789
3 634567891 3451212
4 973456790 3314567
5 672345678 2335689
```

Рисунок 7

## РЕШЕНИЕ ЗАДАНИЯ 2

1. Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем необходимое число переменных и констант. Одна целочисленная константа необходима для хранения длины массива. В нашем примере длина массива будет десять. Далее объявляем целочисленный массив, используя созданную ранее целочисленную константу для задания размера массива, и две целочисленные переменные: одну для хранения переставляемого элемента массива и вторую — счетчик количества перестановок. Организуем цикл для ввода элементов массива с консоли.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Home task #9.2.2\n";

    const int n = 10;
    int mass[n], temp, nShift;

    cout << "Please, enter elements of the array.\n";
    for (int i = 0; i < n; i++)
    {
        cin>> mass[i];
    }
}
```

```
cout << "\n";  
return 0;  
}
```

## 2. Реализуем классический алгоритм сортировки «пузырьком».

Будем анализировать массив от начала до конца, меняя местами неотсортированные соседние элементы.

В результате первого прохода подобным образом на последнее место в массиве «всплывёт» минимальный элемент. Далее повторяем обход неотсортированной части массива (от первого элемента до предпоследнего, т.к. последний уже на своем месте) и меняем по пути неотсортированных соседей.

Количество проходов равно количеству элементов массива за вычетом единицы, т.к. происходит попарное сравнение. Для реализации таких проходов массива организуем внешний цикл, внутри которого располагаем цикл для попарного сравнения элементов.

Длина внутреннего цикла будет на каждом проходе уменьшаться на единицу, т.к. за один проход один «пузырек» всплывает на свое место и число неотсортированных элементов сокращается.

```
#include <iostream>  
  
using namespace std;  
int main()  
{  
    cout << "Home task #9.2.2\n";
```

```
const int n = 10;
int mass[n], temp, nShift;

cout << "Please, enter elements of the array.\n";
for (int i = 0; i < n; i++)
{
    cin>> mass[i];
}

cout << "\n";

for (int i = 1; i < n; ++i)
{
    for (int r = 0; r < n - i; r++)
    {
        if (mass[r] > mass[r + 1])
        {
            temp = mass[r];
            mass[r] = mass[r + 1];
            mass[r + 1] = temp;
            nShift++;
        }
    }
}

return 0;
}
```

3. В начале каждого прохода переменную — счетчик количества перестановок нужно обнулять. Проход реализуется внешним циклом., т.е. перед началом внутреннего цикла (анализ и обработка неотсортированной части массива) нужно обнулять счетчик количества перестановок.

После завершения прохода необходимо проверить, были ли выполнены перестановки. Если их не было (значение переменной — счетчика количества перестановок равно нулю), то значит массив уже отсортирован и следующие проходы не нужны — останавливаем внешний цикл.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Home task #9.2.2\n";

    const int n = 10;
    int mass[n], temp, nShift;

    cout << "Please, enter elements of the array.\n";
    for (int i = 0; i < n; i++)
    {
        cin>> mass[i];
    }

    cout << "\n";

    for (int i = 1; i < n; ++i)
    {
        nShift = 0;
        for (int r = 0; r < n - i; r++)
        {
            if (mass[r] > mass[r + 1])
            {
                temp = mass[r];
```

```
        mass[r] = mass[r + 1];
        mass[r + 1] = temp;
        nShift++;
    }
}

if (nShift == 0)
{
    cout << "The sorting process is stoped at ";
    cout << i << " step\n";
    break;
}

for (int i = 0; i < n; i++)
{
    cout << mass[i] << " ";
}

return 0;
}
```

4. Добавляем цикл для вывода элементов массива после сортировки.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Home task #9.2.2\n";
```



```
const int n = 10;
int mass[n], temp, nShift;

cout << "Please, enter elements of the array.\n";
for (int i = 0; i < n; i++)
{
    cin>> mass[i];
}

cout << "\n";

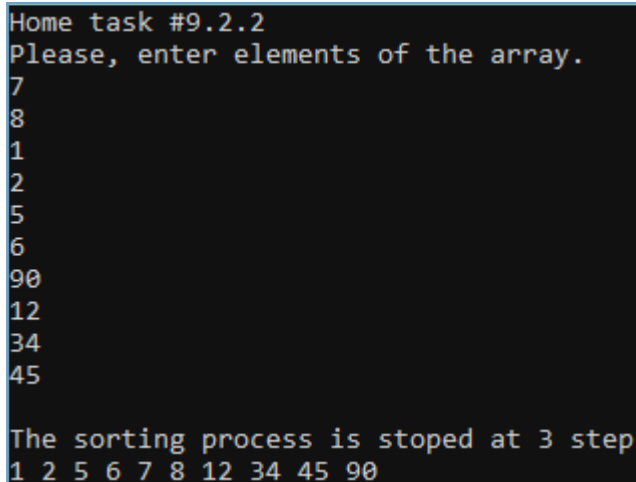
for (int i = 1; i < n; ++i)
{
    nShift = 0;
    for (int r = 0; r < n - i; r++)
    {
        if (mass[r] > mass[r + 1])
        {
            temp = mass[r];
            mass[r] = mass[r + 1];
            mass[r + 1] = temp;
            nShift++;
        }
    }
    if (nShift == 0)
    {
        cout << "The sorting process is stoped at "
        cout << i << " step\n";
        break;
    }
}

for (int i = 0; i < n; i++)
{
```

```
        cout << mass[i] << " ";  
    }  
  
    return 0;  
}
```

Результаты работы программы (в консоли):

- Тест 1 — Часть исходного массива уже отсортирована



```
Home task #9.2.2  
Please, enter elements of the array.  
7  
8  
1  
2  
5  
6  
90  
12  
34  
45  
  
The sorting process is stoped at 3 step  
1 2 5 6 7 8 12 34 45 90
```

Рисунок 8

- Тест 2 — Исходный массив отсортирован в обратном порядке, т.е. нужно переставить все элементы (остановки не будет)

```
Home task #9.2.2
Please, enter elements of the array.
12
11
10
9
8
7
6
5
4
3
3 4 5 6 7 8 9 10 11 12
```

Рисунок 9