Структурированные типы данных

Структура данных — это некоторая структура, содержащая данные.

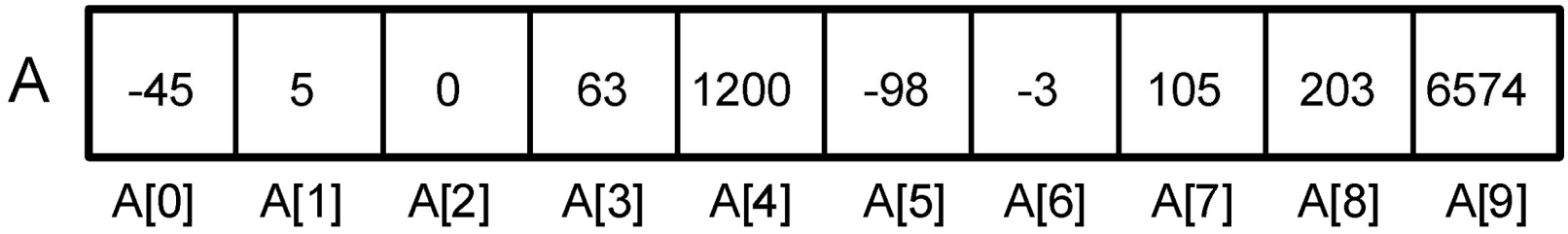
Различают динамические и статические типы данных.

Статическая структура данных имеет фиксированный тип и размер. Для ее изменения программист должен изменить исходный код программы. В то время как размер (а иногда и тип) *динамической структуры* данных может изменяться в процессе выполнения программы.

Массивы

Массив представляет собой упорядоченный набор однотипных элементов. Массивы бывают одномерными и многомерными.

Одномерные массивы (вектора) — это конечная именованная последовательность элементов. Элементы массива занимают в памяти один непрерывный участок памяти и располагаются последовательно друг за другом. Для доступа к отдельному элементу массива указывают имя массива и номер позиции отдельного элемента массива (***индекс***). Индекс должен быть целым числом или целым выражением. Индексация элементов массива начинается с нуля. На рисунке показан вектор A целых чисел.



Объявление вектора

Массивы занимают область в памяти. При объявлении вектора указывают тип каждого элемента и количество элементов. При этом компилятор резервирует соответствующий объем памяти.

Объявление одномерного массива:

```
<тип элемента> <имя массива>[<кол-во элементов>];
```

Например: `int A[10];`

```
float B[3],C[100];
double D[20];
```

Инициализация вектора

Инициализация одномерного массива означает присвоение начальных значений его элементам при объявлении. Массивы можно инициализировать списком значений или выражений.

Например:

```
int days[12]={31,28,31,30,31,30,31,31,30,31,30,31};
```

Если элементов в массиве больше, чем инициализаторов, элементы для которых значения не указаны обнуляются. Массив можно инициализировать списком без указания в скобках длины. При этом длина массива определяется количеством инициализаторов.

Например: `char code[]={'a','b','c'};`

Если массив явно не проинициирован, то внешние и статические массивы инициализируются нулями. Автоматические массивы после объявления ничем не инициализируются и содержат неизвестную информацию.

Передача массивов в качестве параметров функции

При использовании в качестве параметра массива в функцию передается указатель на его первый элемент, т.е., массив всегда передается по адресу. При этом информация о количестве элементов массива теряется, и следует передавать его размерность через отдельный параметр.

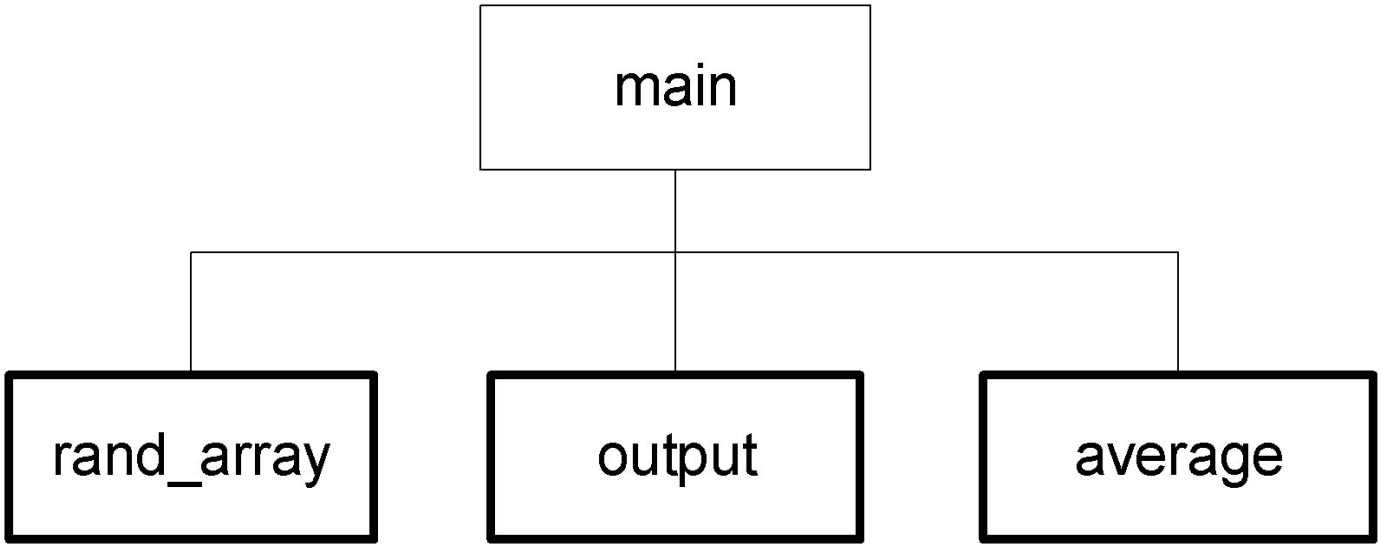
Например:

```
//функция ввода элементов вектора
void input_array(int mas[],int kol_el)
{
        for (int i=0;i<kol_el;i++)
            {     cout<<"Input "<<i<<" element-->" ;
                  cin>>mas[i];
            }
}
```

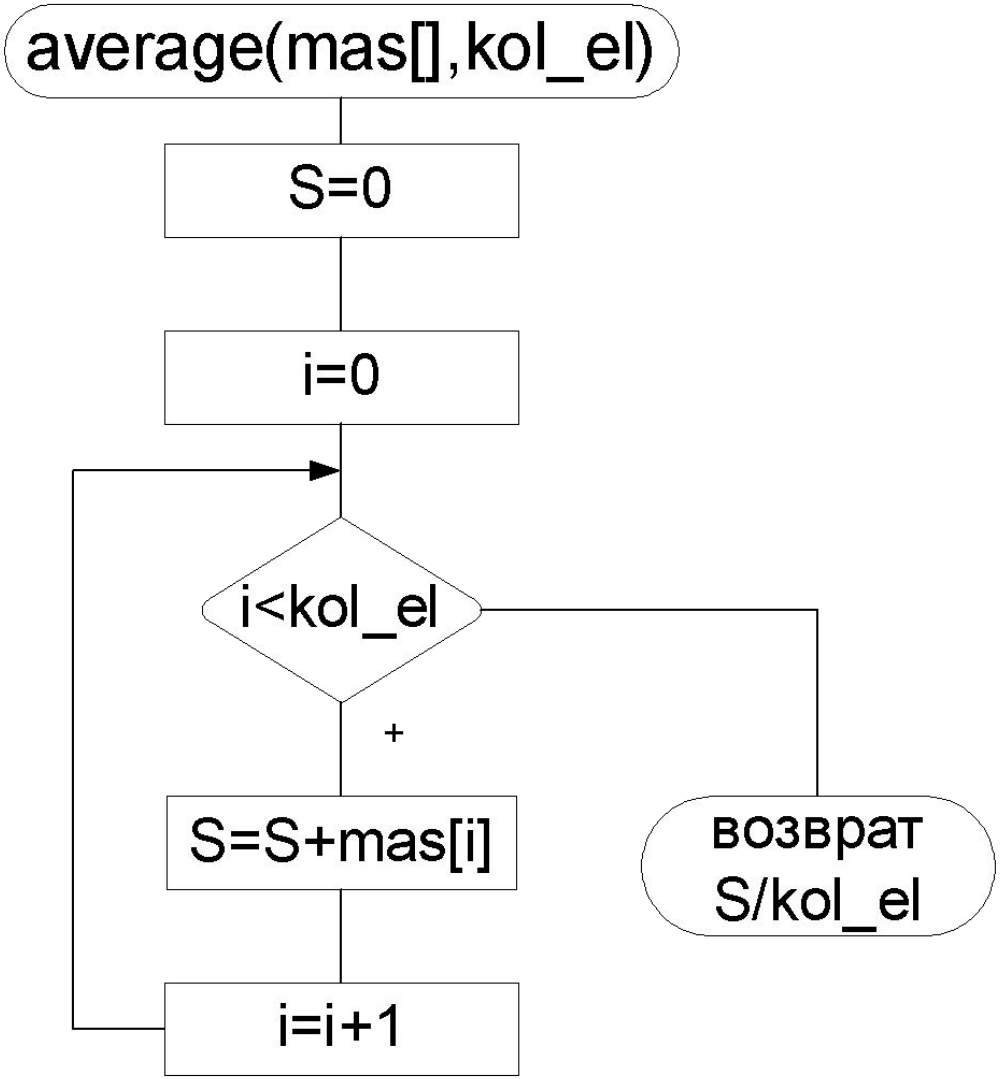
Пример EX18_1.cpp

Сформировать с помощью датчика случайных чисел целочисленный вектор заданного размера (не более 10), вывести его на экран. Найти среднее арифметическое элементов вектора.

Структурная схема программы:



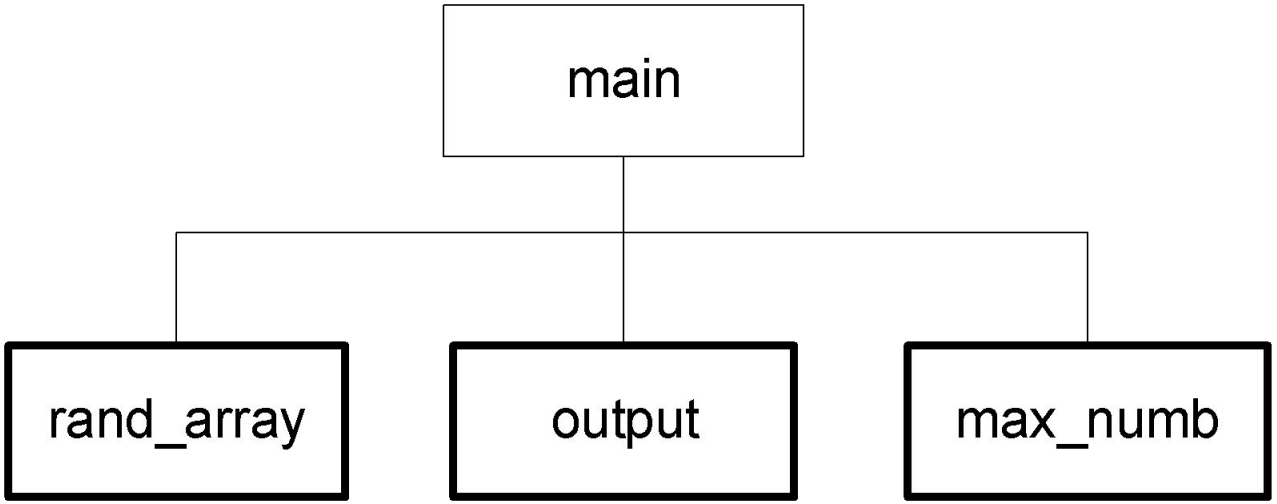
Пример EX18_1.cpp
блок-схема функции average



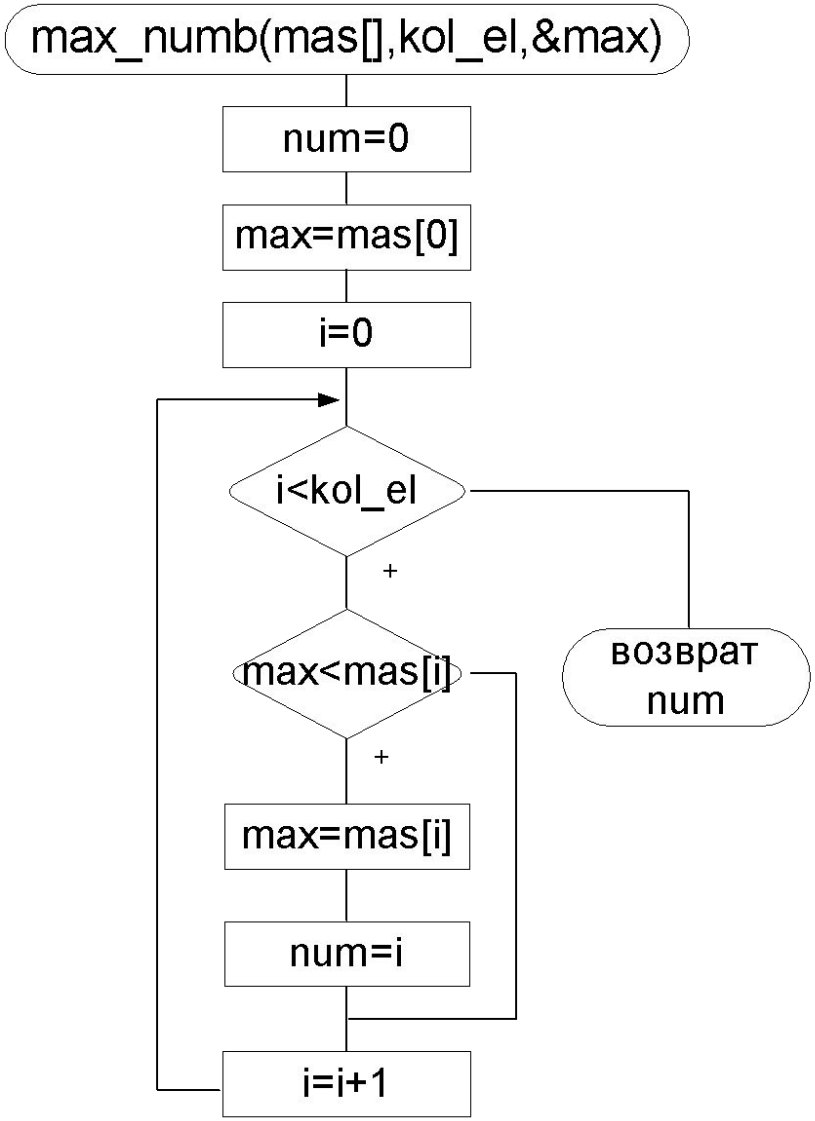
Пример EX19_1.cpp

Сформировать с помощью датчика случайных чисел целочисленный вектор заданного размера (не более 10), вывести его на экран. Найти максимальный элемент и его номер.

Структурная схема программы:



Пример EX19_1.cpp
блок-схема функции max_numb



Алгоритмы сортировки элементов вектора

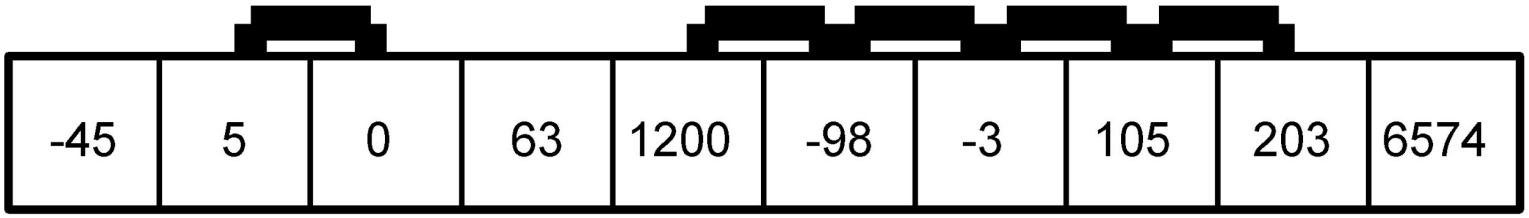
Сортировка данных (размещение данных в определенном порядке) одна из важнейших задач.

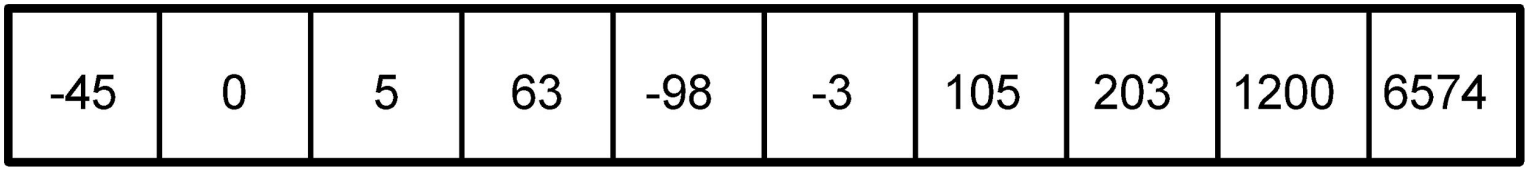
Пузырьковая сортировка: наименьшее значение постепенно «всплывает», продвигаясь к началу массива. Метод требует несколько подходов к вектору. При каждом подходе сравнивается пара следующих друг за другом элементов. При необходимости элементы меняются местами и фиксируется факт обмена. Сортировка заканчивается когда обменов не было.

Пузырьковая сортировка

A	-45	5	0	63	1200	-98	-3	105	203	6574
	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

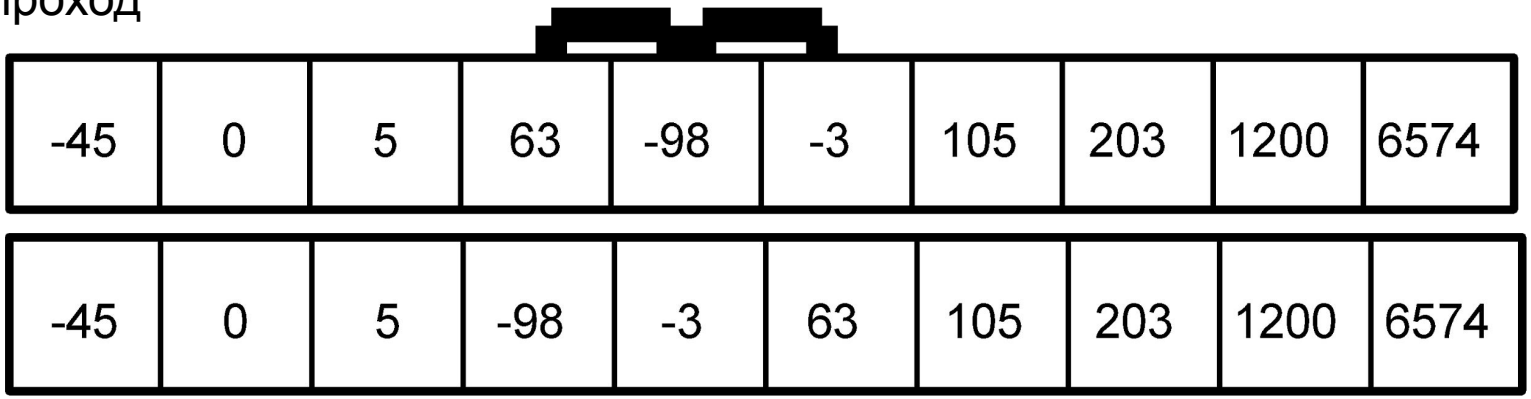
1 проход



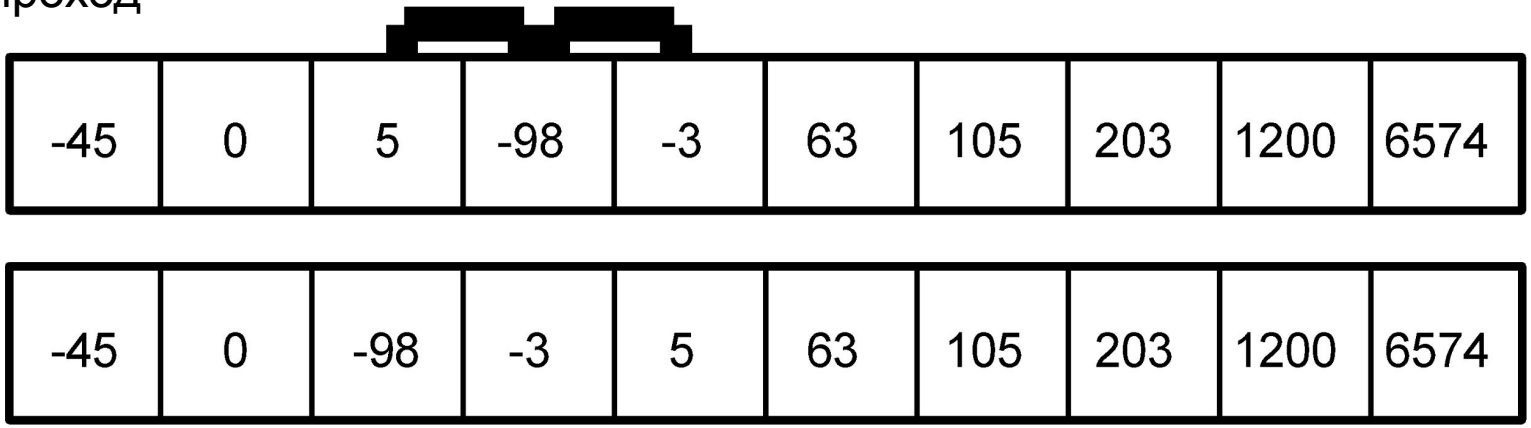


Пузырьковая сортировка

2 проход

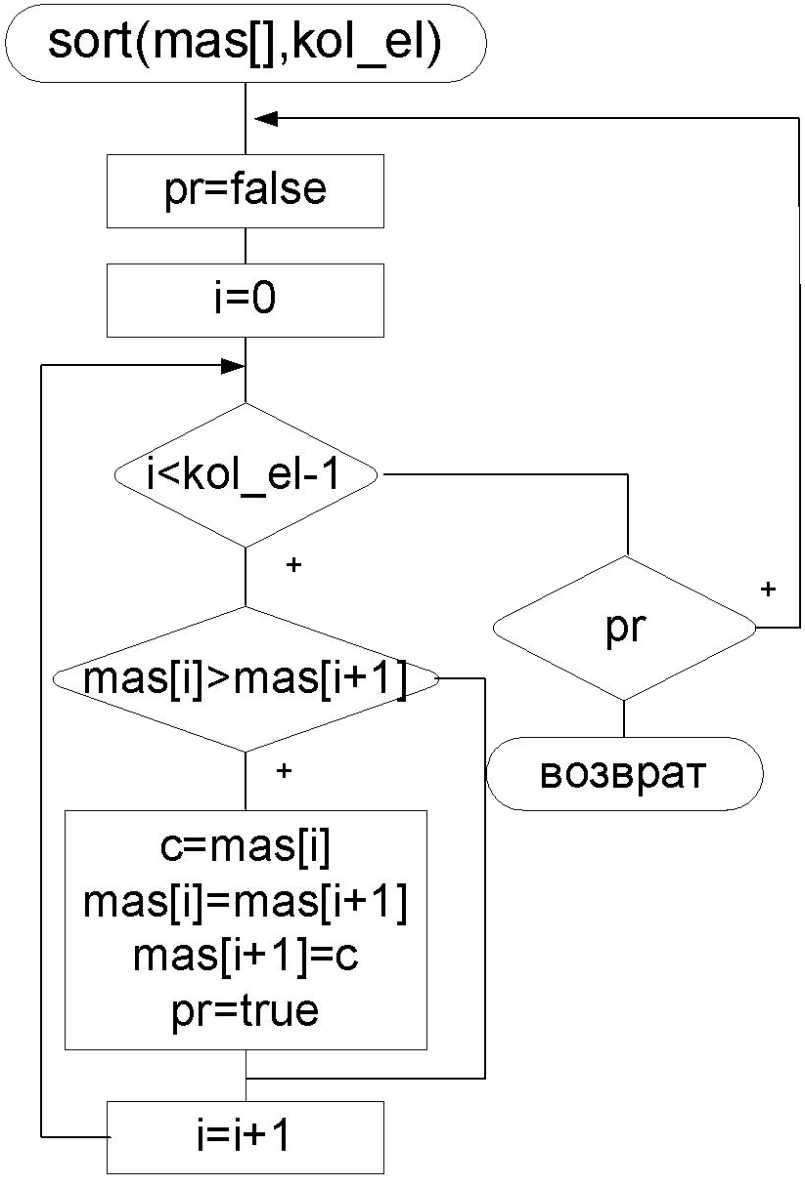


3 проход



И т.д. всего получится 6 проходов

Пример EX20_1.cpp блок-схема алгоритма пузырьковой сортировки



Сортировка методом выбора

Просматриваются все элементы вектора начиная с первого и выбирается наименьшее значение. Наименьшее значение меняется местами с первым элементом. Просмотр вектора и выбор наименьшего начинается со второго элемента. Найденное наименьшее значение меняется местами со вторым элементом и т. д. пока не будут просмотрены все элементы.

A

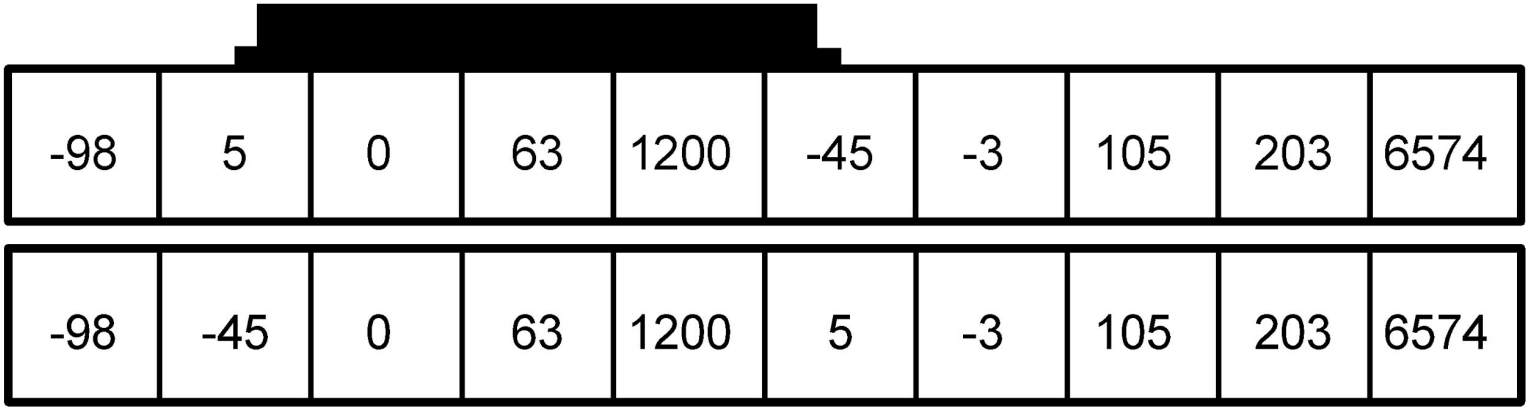
-45	5	0	63	1200	-98	-3	105	203	6574
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

Начинаем с 1-го элемента

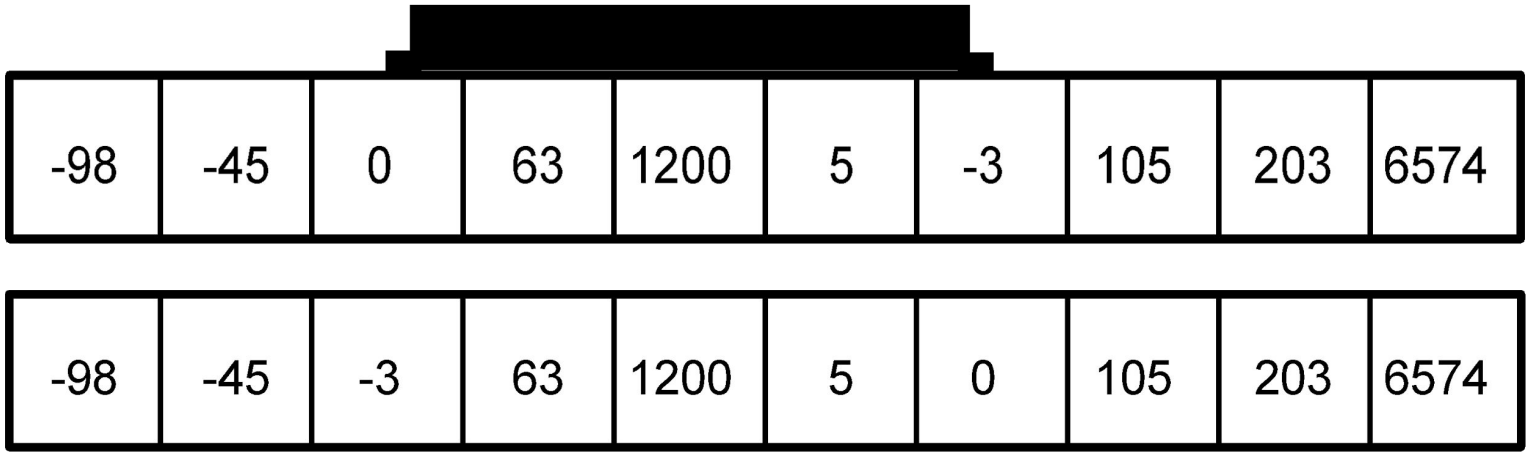
-45	5	0	63	1200	-98	-3	105	203	6574
-98	5	0	63	1200	-45	-3	105	203	6574

Сортировка методом выбора

Начинаем со 2-го элемента



Начинаем с 3-го элемента



Пример EX21_1.cpp блок-схема алгоритма сортировки методом выбора