Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе № 15**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Методы внутренней сортировки массивов: быстрая сортировка, сортировка подсчётом

Вариант 21

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

Изучить принцип работы быстрых алгоритмов Шелла и Хоара.

**Постановка задачи**

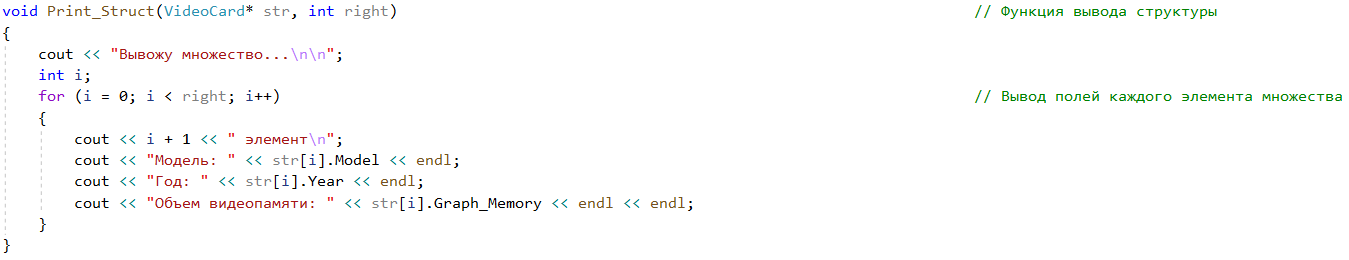
Создать структуру данных. Провести сортировку во множестве по ключу, используя алгоритмы быстрых поисков Шелла и Хоара.

**Анализ задачи**

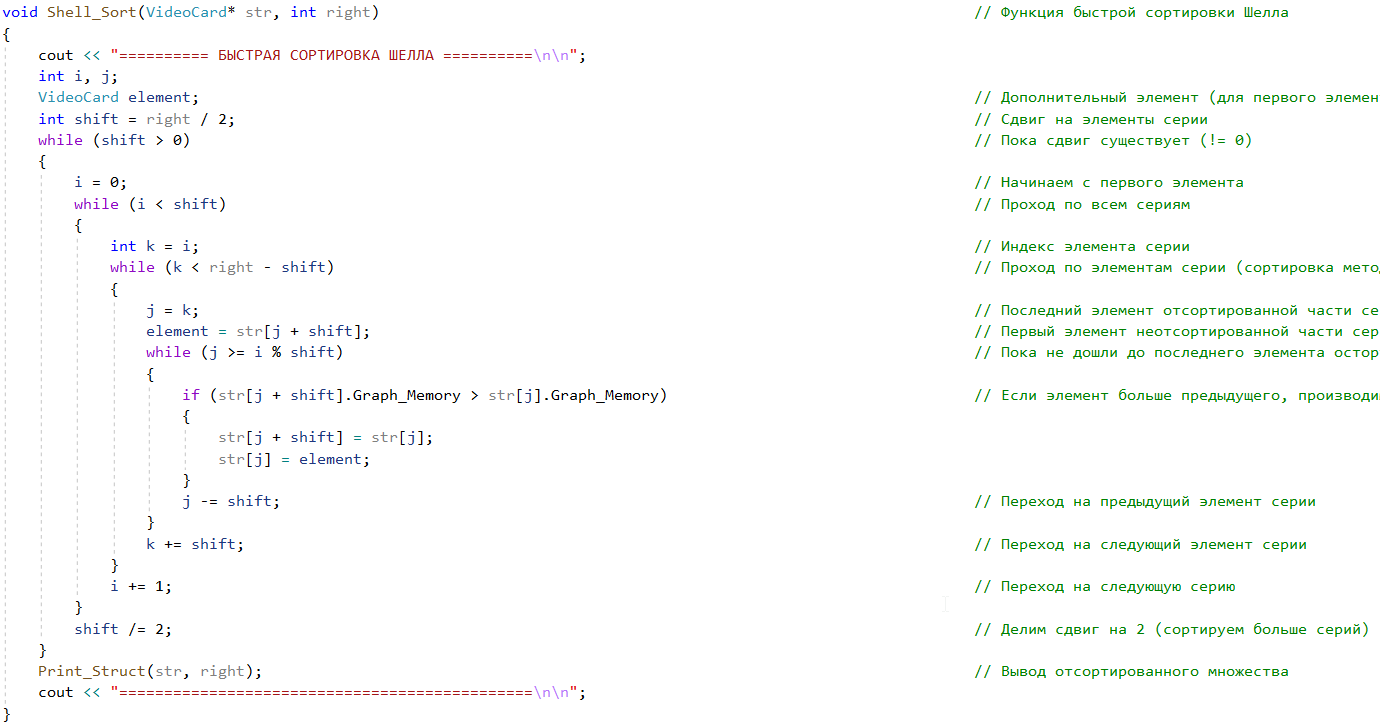
1. Для решения задачи необходимо:
   1. Реализовать функцию создания множества.



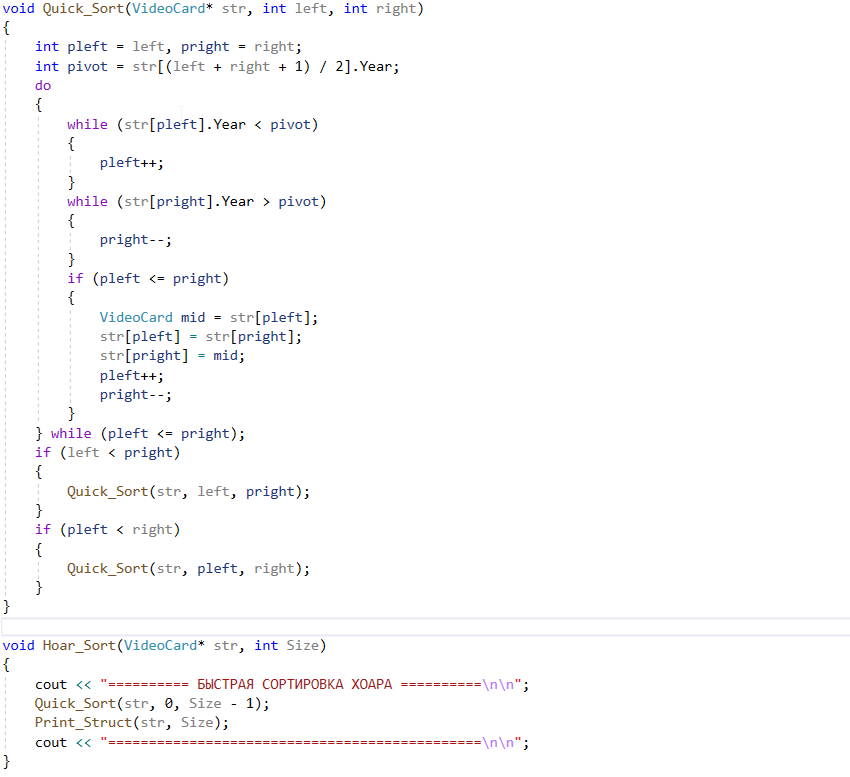
* 1. Разработать функцию вывода множества через консоль.



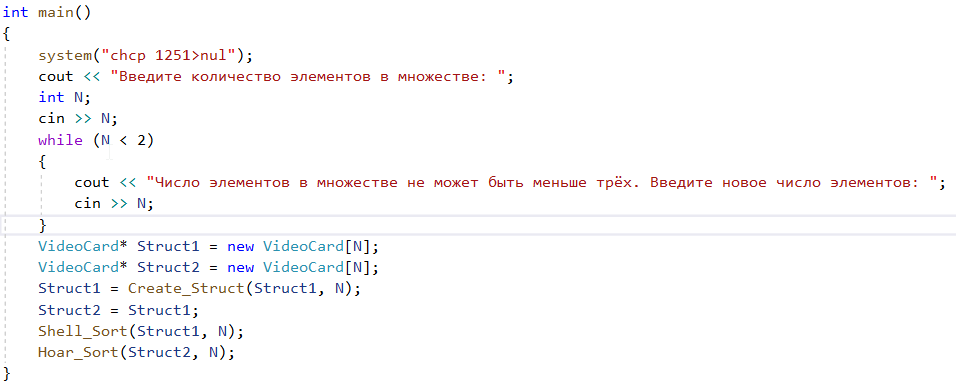
* 1. Разработать функцию метода быстрой сортировки Шелла.



* 1. Разработать функцию метода быстрой сортировки Хоара.



* 1. Реализовать применение этих функций в главной функции.



1. В ходе работы были использованы типы данных:
   1. Для функции Create\_Struct используются следующие аргументы:
   2. Указатель на VideoCard: пустое множество элементов.
   3. Тип int: размерность множества.

Сама функция имеет тип указателя на VideoCard и возвращает заполненное множество.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_ThVbNjSGdr.png

* 1. Для функции Print\_Struct используются следующие аргументы:
  2. Указатель на VideoCard: множество элементов.
  3. Тип int: размерность множества.

Сама функция имеет тип void, поскольку при работе функции не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_WNOGBzFp1G.png

* 1. Для функции Quick\_Sort используются следующие аргументы:
  2. Указатель на VideoCard: неотсортированное множество элементов.
  3. Тип int: левая граница подмножества.
  4. Тип int: правая граница подмножества.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_JOIONrglQA.png

* 1. Для функции Hoar\_Sort используются следующие аргументы:
  2. Указатель на VideoCard: неотсортированное множество элементов.
  3. Тип int: размерность множества.

Сама функция имеет тип void, поскольку при работе функции не нужно возвращать значение.

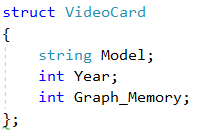
C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_eGdk2N14yX.png

* 1. Для функции Shell\_Sort используются следующие аргументы:
  2. Указатель на VideoCard: неотсортированное множество элементов.
  3. Тип int: размерность множества.

Сама функция имеет тип bool и возвращает истину, если параметр при полном выполнении функции равен 7.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_RWjFjQLTW8.png

1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы данные были записаны в структуру данных с тремя полями для хранения данных.



1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных через консоль реализован с помощью оператора cin и функции getline.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_JXRYwtKSlw.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_0Uzl50RklT.png

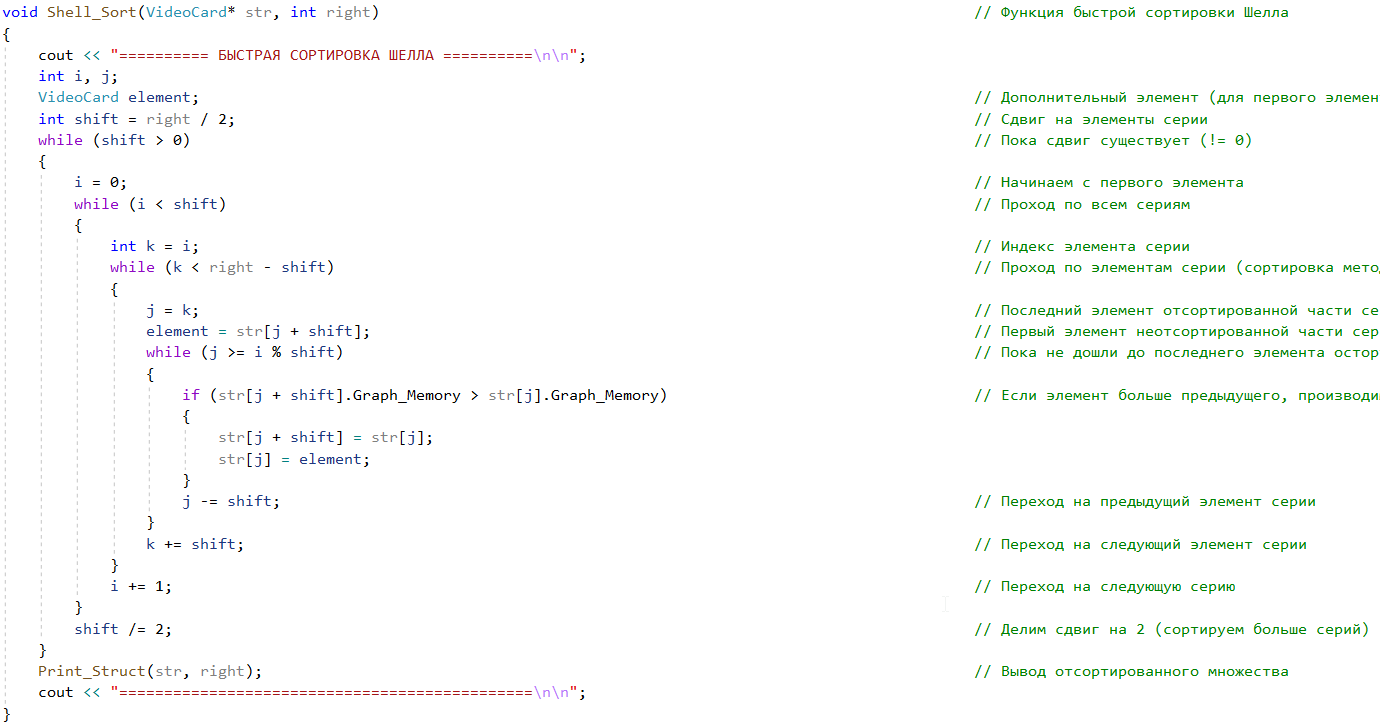
C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_Ix7JnukgS4.png

* 1. Вывод данных на консоль реализован с помощью оператора cout.

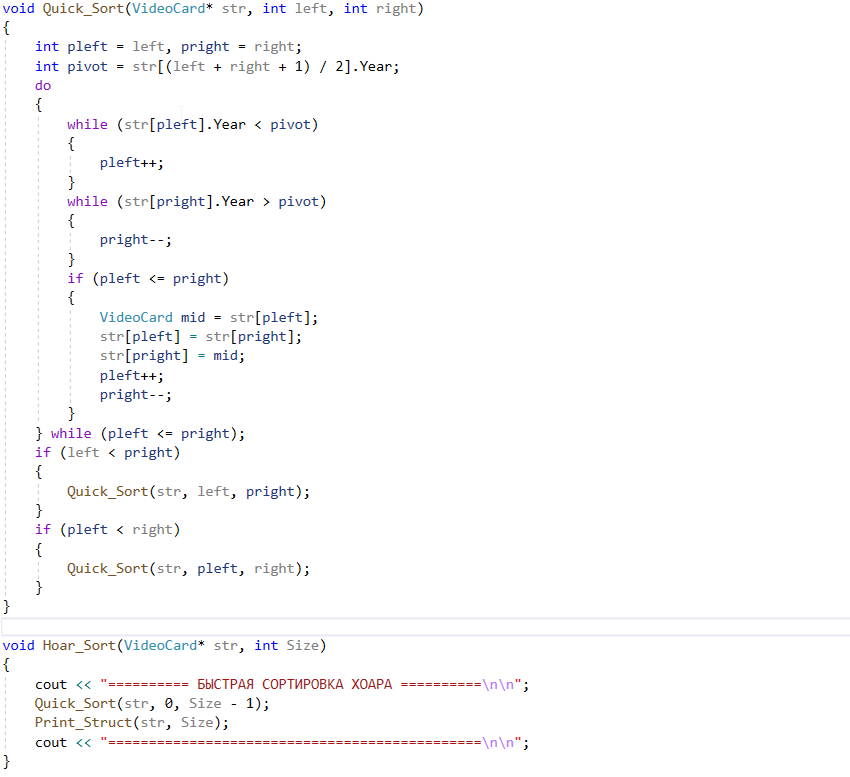
C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_LNOYbqcbhm.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_8kTwcidE1U.png

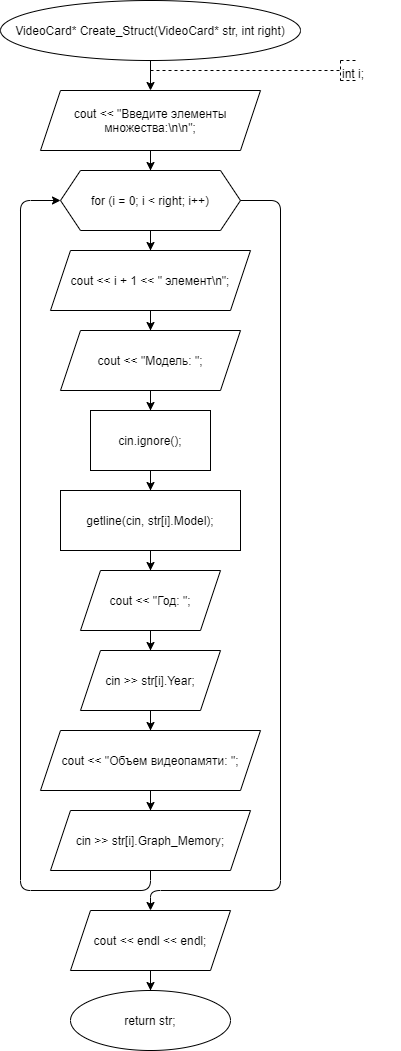
1. Поставленные задачи решены следующими действиями:
   1. Для реализации Метода Быстрой Сортировки Шелла была реализована функция Shell\_Sort. Реализация метода похожа на метод вставки, но дополнительно проверяется значение сдвига элементов. Это сдвиг характеризует серии. Алгоритм выполняется, пока сдвиг элементов не равен 0.

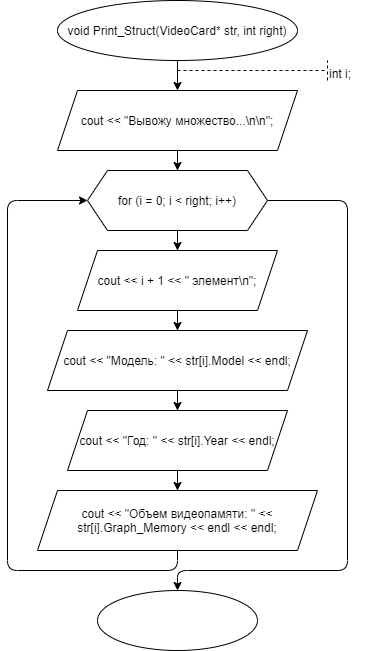


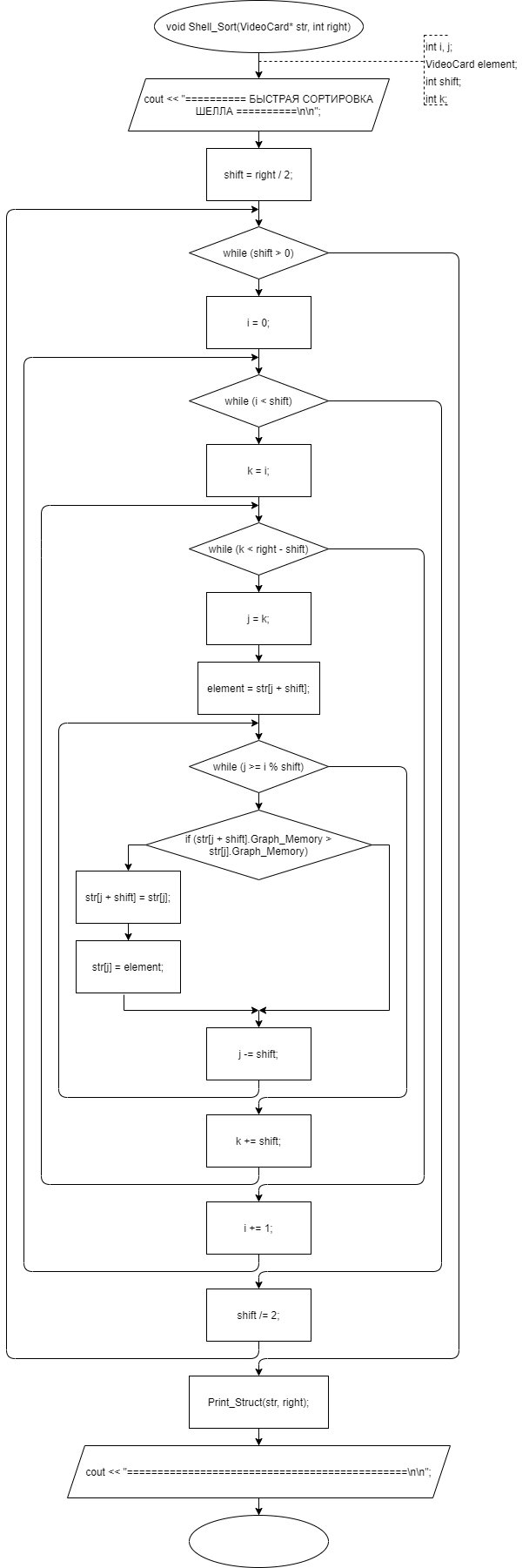
* 1. Для реализации Метода Быстрой Сортировки Хоара были реализованы функции Hoar\_Sort и Quick\_Sort. Hoar\_Sort единожды вызывает функцию Quick\_Sort. При работе функции Quick\_Sort находятся парно индекс элементов, стоящих не на своём месте (т. е. неотсортированы) и меняются местами. Проверяется индекс найденных элементов.

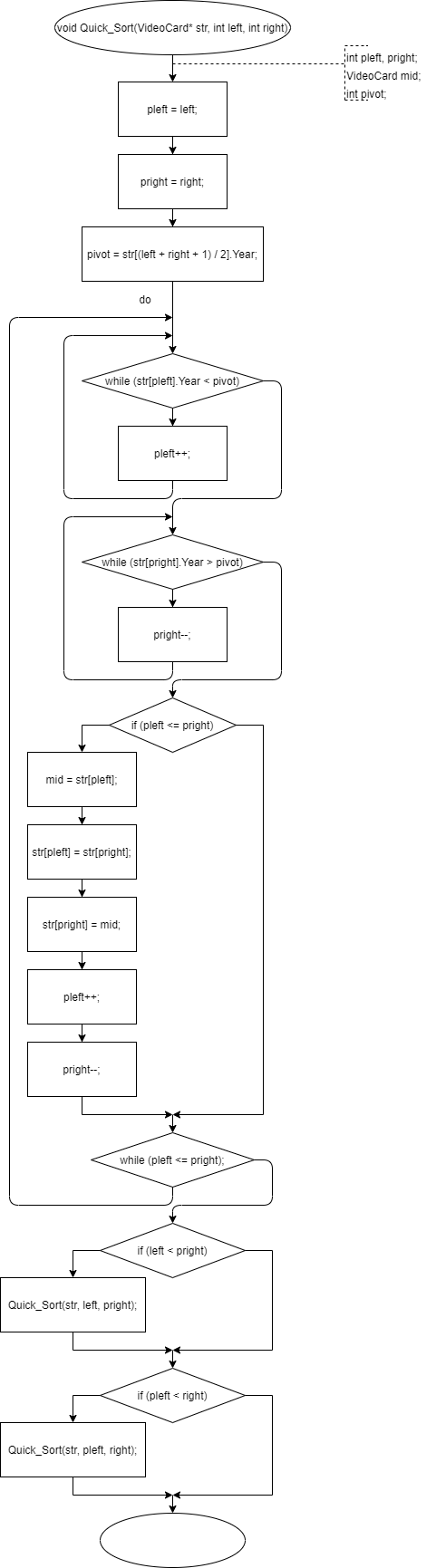


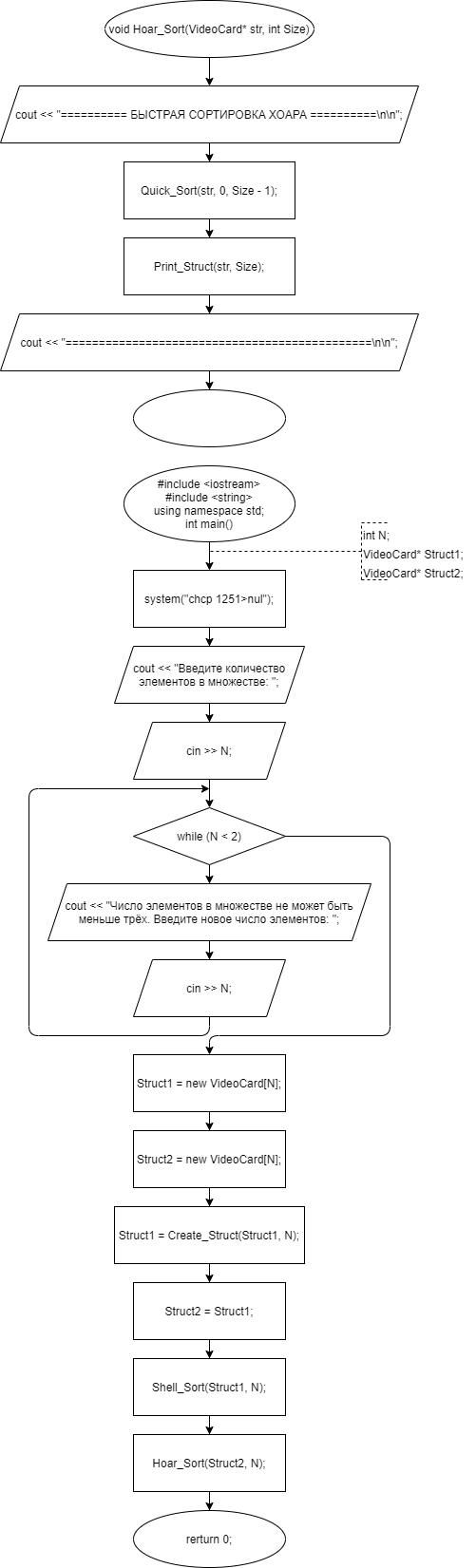
**Блок-схема программы**











**Реализация задачи на языке С++**

#include <iostream>

#include <string>

using namespace std;

struct VideoCard

{

string Model;

int Year;

int Graph\_Memory;

};

VideoCard\* Create\_Struct(VideoCard\* str, int right)

{

cout << "Введите элементы множества:\n\n";

int i;

for (i = 0; i < right; i++)

{

cout << i + 1 << " элемент\n";

cout << "Модель: ";

cin.ignore();

getline(cin, str[i].Model);

cout << "Год: ";

cin >> str[i].Year;

cout << "Объем видеопамяти: ";

cin >> str[i].Graph\_Memory;

}

cout << endl << endl;

return str;

}

void Print\_Struct(VideoCard\* str, int right) {

cout << "Вывожу множество...\n\n";

int i;

for (i = 0; i < right; i++)

{

cout << i + 1 << " элемент\n";

cout << "Модель: " << str[i].Model << endl;

cout << "Год: " << str[i].Year << endl;

cout << "Объем видеопамяти: " << str[i].Graph\_Memory << endl << endl;

}

}

void Shell\_Sort(VideoCard\* str, int right)

{

cout << "========== БЫСТРАЯ СОРТИРОВКА ШЕЛЛА ==========\n\n";

int i, j;

VideoCard element;

int shift = right / 2;

while (shift > 0)

{

i = 0;

while (i < shift)

{

int k = i;

while (k < right - shift)

{

j = k;

element = str[j + shift];

while (j >= i % shift)

{

if (str[j + shift].Graph\_Memory > str[j].Graph\_Memory)

{

str[j + shift] = str[j];

str[j] = element;

}

j -= shift;

}

k += shift;

}

i += 1;

}

shift /= 2;

}

Print\_Struct(str, right);

cout << "==============================================\n\n";

}

void Quick\_Sort(VideoCard\* str, int left, int right)

{

int pleft = left, pright = right;

int pivot = str[(left + right + 1) / 2].Year;

do

{

while (str[pleft].Year < pivot)

{

pleft++;

}

while (str[pright].Year > pivot)

{

pright--;

if (pleft <= pright)

{

VideoCard mid = str[pleft];

str[pleft] = str[pright];

str[pright] = mid;

pleft++;

pright--;

}

} while (pleft <= pright);

if (left < pright)

{

Quick\_Sort(str, left, pright);

}

if (pleft < right)

{

Quick\_Sort(str, pleft, right);

}

}

void Hoar\_Sort(VideoCard\* str, int Size)

{

cout << "========== БЫСТРАЯ СОРТИРОВКА ХОАРА ==========\n\n";

Quick\_Sort(str, 0, Size - 1);

Print\_Struct(str, Size);

cout << "==============================================\n\n";

}

int main()

{

system("chcp 1251>nul");

cout << "Введите количество элементов в множестве: ";

int N;

cin >> N;

while (N < 2)

{

cout << "Число элементов в множестве не может быть меньше трёх. Введите новое число элементов: ";

cin >> N;

}

VideoCard\* Struct1 = new VideoCard[N]; VideoCard\* Struct2 = new VideoCard[N]; Struct1 = Create\_Struct(Struct1, N);

Struct2 = Struct1; Shell\_Sort(Struct1, N);

Hoar\_Sort(Struct2, N);

}

**Скриншоты результатов работы программы**

