Отчет по программированию на С++  
Билет №3  
  
Задача 1

Задание: Создайте класс Phone, который содержит переменные number, model и weight.

а) Создайте три экземпляра этого класса.

б) Выведите на консоль значения их переменных.

в) Добавить в класс Phone методы: receiveCall, имеет один параметр – имя звонящего. Выводит на консоль сообщение “Звонит {name}”. Метод getNumber – возвращает номер телефона. Вызвать эти методы для каждого из объектов.

г) Добавить конструктор в класс Phone, который принимает на вход три параметра для инициализации переменных класса - number, model и weight.

д) Добавить конструктор, который принимает на вход два параметра для инициализации переменных класса - number, model.

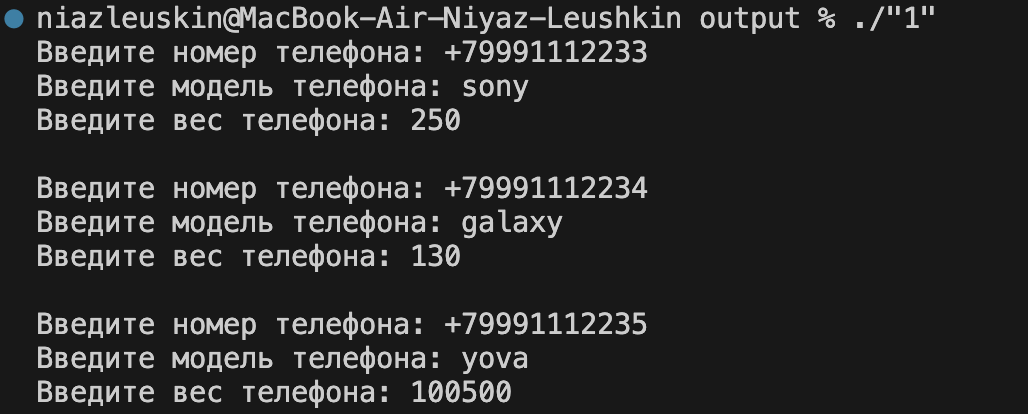
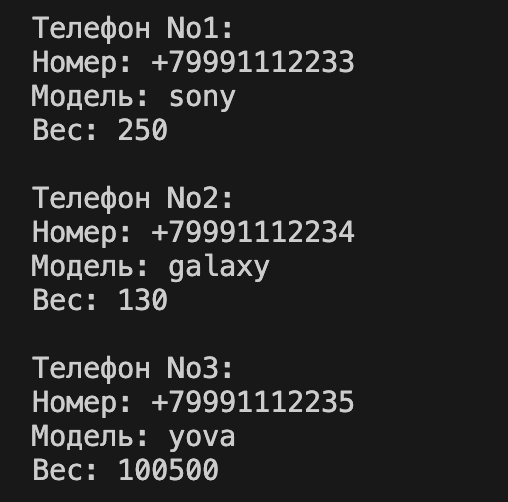
е) Добавить конструктор без параметров.

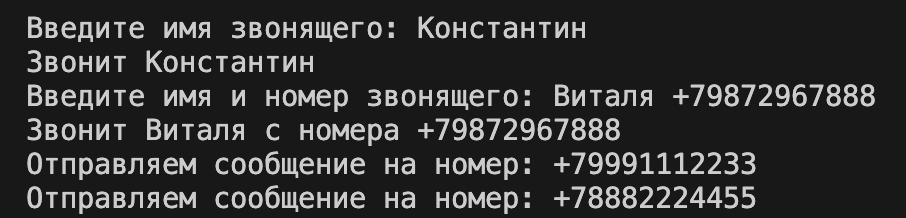
ж) Вызвать из конструктора с тремя параметрами конструктор с двумя.

з) Добавьте перегруженный метод receiveCall, который принимает два параметра - имя звонящего и номер телефона звонящего. Вызвать этот метод.

и) Создать метод sendMessage с аргументами переменной длины. Данный метод принимает на вход номера телефонов, которым будет отправлено сообщение. Метод выводит на консоль номера этих телефонов.

Для использования программы 1.cpp нужно:

1. Запустить программу командой:  
   g++ 1.cpp -o 1 && ./1
2. Ввести данные 3-х телефонов: номер, модель, вес. Пример: 
3. Программа выведет информацию о созданных объектах класса Phone: 
4. Введите имя звонящего, чтобы вызвать метод receiveCall.
5. Введите имя и номер звонящего, чтобы вызвать перегруженный метод receiveCall.
6. Будут отправлены сообщения на заданные номера с помощью метода sendMessage:



Сам код программы 1.cpp:

#include <iostream>

#include <string>

using namespace std;

class Phone

{

public:

string number;

string model;

int weight;

Phone(string num, string mod, int w)

{

number = num;

model = mod;

weight = w;

}

Phone(string num, string mod)

{

Phone(num, mod, 0);

}

Phone()

{

Phone("", "", 0);

}

void receiveCall(string name)

{

cout << "Звонит " << name << endl;

}

string getNumber()

{

return number;

}

void receiveCall(string name, string num)

{

cout << "Звонит " << name << " с номера " << num << endl;

}

void sendMessage(int count, ...)

{

va\_list args;

va\_start(args, count);

for (int i = 0; i < count; i++)

{

string \*num = va\_arg(args, string \*);

cout << "Отправляем сообщение на номер: " << \*num << endl;

}

va\_end(args);

}

};

int main()

{

Phone phones[3];

for (int i = 0; i < 3; i++)

{

string number, model;

int weight;

cout << "Введите номер телефона: ";

cin >> number;

cout << "Введите модель телефона: ";

cin >> model;

cout << "Введите вес телефона: ";

cin >> weight;

phones[i] = Phone(number, model, weight);

cout << endl;

}

for (int i = 0; i < 3; i++)

{

cout << "Телефон No" << i + 1 << ":" << endl;

cout << "Номер: " << phones[i].getNumber() << endl;

cout << "Модель: " << phones[i].model << endl;

cout << "Вес: " << phones[i].weight << endl;

cout << endl;

}

string name, num;

cout << "Введите имя звонящего: ";

cin >> name;

phones[0].receiveCall(name);

cout << "Введите имя и номер звонящего: ";

cin >> name >> num;

phones[1].receiveCall(name, num);

string num1 = "+79991112233";

string num2 = "+78882224455";

phones[2].sendMessage(2, &num1, &num2);

return 0;

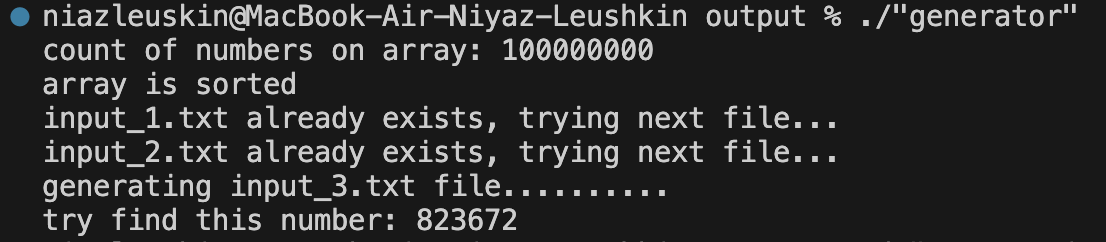
}

Задача 2

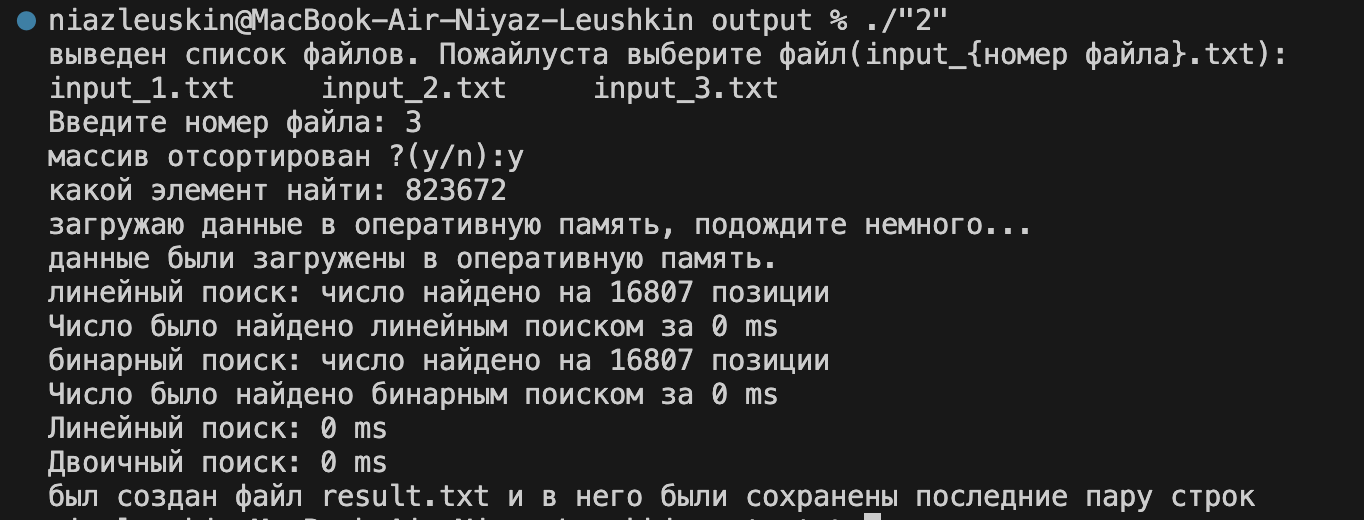
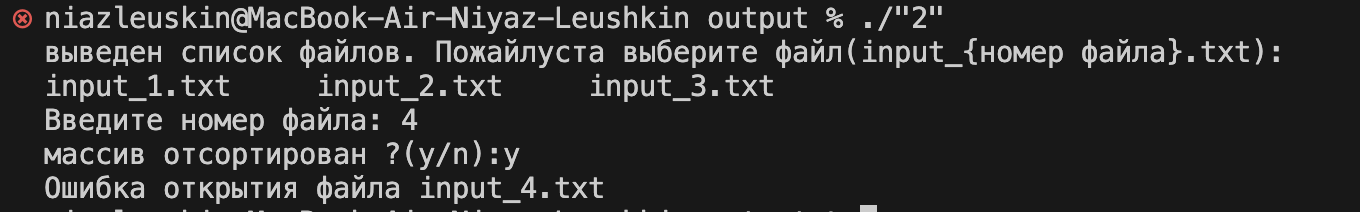
Задание: Напишите метод, который проверяет, входит ли в массив заданный элемент или нет. Используйте перебор и двоичный поиск для решения этой задачи. Сравните время выполнения обоих решений для больших массивов (например, 100000000 элементов).

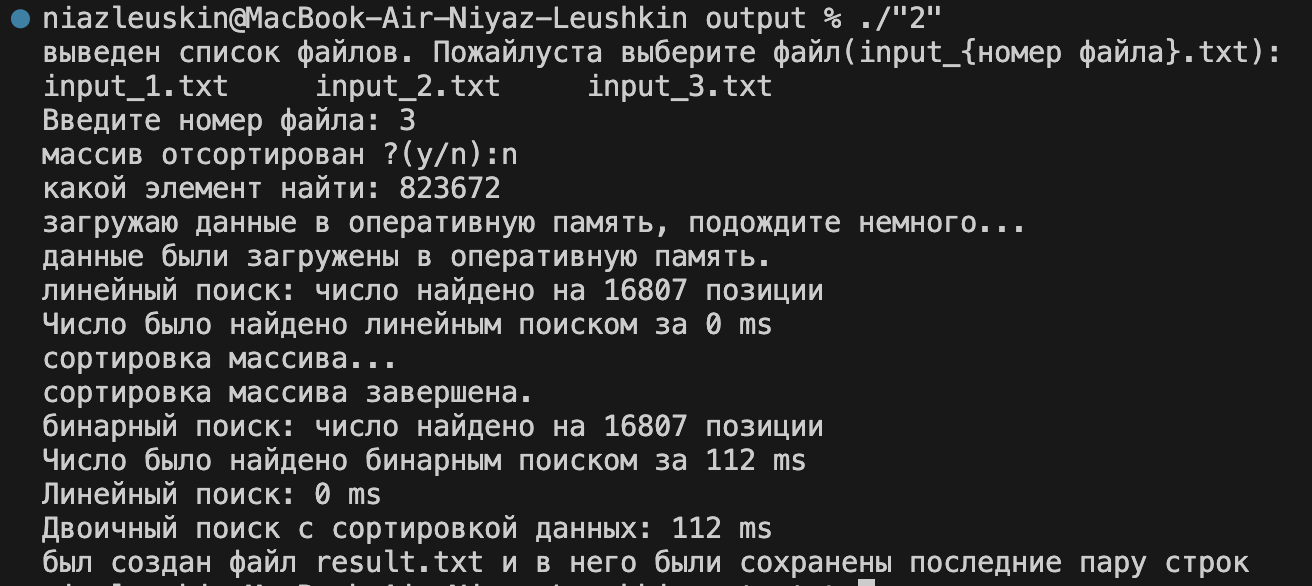
**В решении продемонстрировать  
- использование статических методов   
- обработку исключительных ситуаций   
- чтение данных из файлов, запись в файл результатов вычислений**

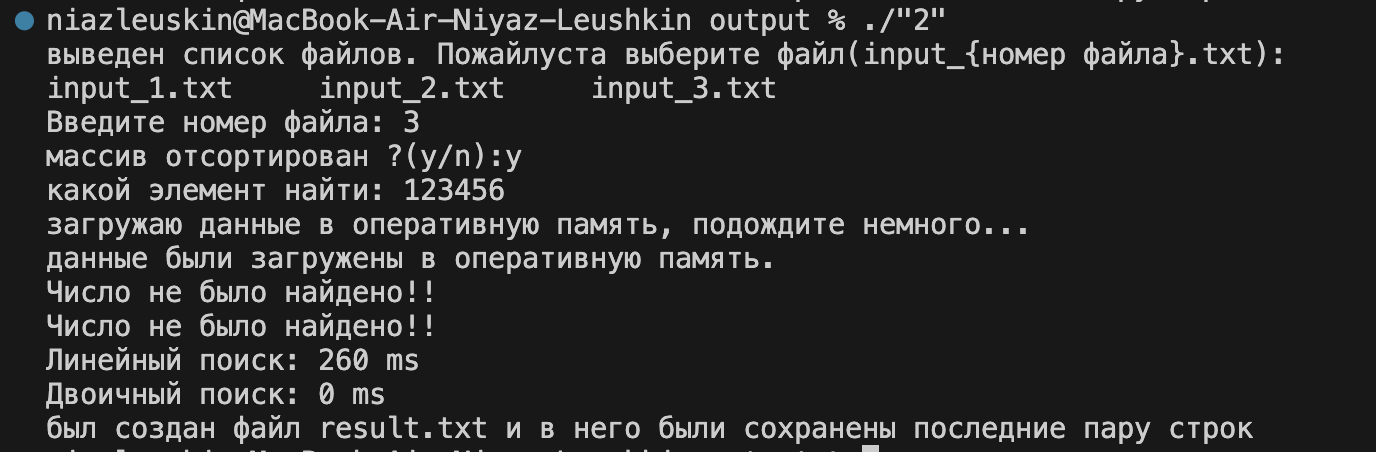
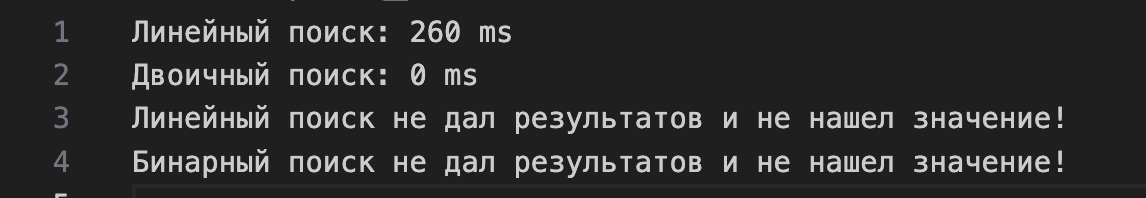
Для запуска программы 2.cpp:

1. Собрать исходники командой:  
   g++ generator.cpp -o generator && g++ 2.cpp -o 2
2. Запустить генератор данных ./generator, указав размер массива и сортировку(в исходниках, по умолчанию размер массива равен 100млн и сортировка включена). Будет создан файл input\_N.txt.  
   Где N это номер файла. Если файл с номером 1 существует, будет создан файл с номером 2 и тд. Предложит найти число, которое 100% есть в массиве и было случайно выбрано.  
   
3. Запустить программу ./2, выбрав сгенерированный файл.
4. Введите элемент для поиска.

Программа выполнит линейный и бинарный поиск, выведет результаты на экран и в файл result.txt.

  
  
Если вдруг файл input\_N.txt не был найден:  


Если вдруг файл не был отсортирован:  


Если вдруг элемента нет в массиве:  
  
  
так же, содержимое файла result.txt:  
  
  
Код generator.cpp:

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

#include <ctime>

using namespace std;

int main()

{

const int n = 100000000;

const bool isSort = true;

cout << "count of numbers on array: " << n << endl;

cout << (isSort ? "array is sorted" : "array is not sorted") << endl;

int file\_num = 1;

std::string fname = "";

while (true)

{

fname = "input\_" + std::to\_string(file\_num) + ".txt";

// Проверка на существование файла

std::ifstream fin(fname);

if (fin)

{

std::cout << fname << " already exists, trying next file..." << std::endl;

file\_num++;

continue;

}

else

{

break;

}

}

// Файл не существует, можно открыть для записи

std::ofstream fout(fname);

cout << "generating " << fname << " file";

cout.flush();

// запись размера в начало

fout << n << "\n";

// элемент для поиска в диапазоне

int ix = rand() % n;

int x = 1000;

// генерация отсортированных данных

srand(time(nullptr));

int prev = 0;

int delay = n / 10;

if (isSort)

{

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

{

std::uint64\_t next = prev + rand() % 100;

fout << next << "\n";

if (i % delay == 0)

{

cout << ".";

cout.flush();

}

if (i == ix)

{

x = next;

}

prev = next;

}

}

else

{

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

{

std::uint64\_t next = rand() % 1000000;

fout << next << "\n";

if (i % delay == 0)

{

cout << ".";

cout.flush();

}

if (i == ix)

{

x = next;

}

}

}

cout << endl;

// fout << x << "\n";

cout << "try find this number: " << x << endl;

fout.close();

return 0;

}

Код 2.cpp:

#include <iostream>

#include <chrono>

#include <fstream>

#include <algorithm>

using namespace std;

// функция линейного поиска

bool linearSearch(vector<int> &arr, int x)

{

for (size\_t i = 0; i < arr.size(); i++)

{

if (arr[i] == x)

{

cout << "линейный поиск: число найдено на " << i << " позиции" << endl;

return true;

}

}

return false;

}

// Функция бинарного поиска

static bool binarySearch(vector<int> &arr, int x)

{

int low = 0;

int high = arr.size() - 1;

while (low <= high)

{

int mid = (low + high) / 2;

if (arr[mid] == x)

{

cout << "бинарный поиск: число найдено на " << mid << " позиции" << endl;

return true;

}

if (arr[mid] < x)

{

low = mid + 1;

}

else

{

high = mid - 1;

}

}

return false;

}

int main()

{

// выводим список файлов

cout << "выведен список файлов. Пожайлуста выберите файл(input\_{номер файла}.txt): " << endl;

system("ls input\*txt");

cout << "Введите номер файла: ";

int file\_num;

cin >> file\_num;

char res;

cout << "массив отсортирован ?(y/n):";

cin >> res;

bool isSorted = (res == 'Y' || res == 'y');

string fname = "input\_" + to\_string(file\_num) + ".txt";

ifstream fin(fname);

if (!fin.is\_open())

{

cout << "Ошибка открытия файла " << fname << endl;

return 1;

}

int x;

cout << "какой элемент найти: ";

cin >> x;

// Загрузка массива из файла

vector<int> arr;

cout << "загружаю данные в оперативную память, подождите немного..." << endl;

cout.flush();

int num;

fin >> num;

while (fin >> num)

{

arr.push\_back(num);

}

cout << "данные были загружены в оперативную память." << endl;

cout.flush();

// линейный поиск

auto start = chrono::high\_resolution\_clock::now();

bool foundLinear = linearSearch(arr, x);

auto end = chrono::high\_resolution\_clock::now();

auto linear\_time = chrono::duration\_cast<chrono::milliseconds>(end - start);

if (foundLinear)

{

cout << "Число было найдено линейным поиском за " << linear\_time.count() << " ms" << endl;

}

else

{

cout << "Число не было найдено!!" << endl;

}

// Двоичный поиск (с сортировкой данных)

start = chrono::high\_resolution\_clock::now();

bool foundBinary;

if (!isSorted)

{

cout << "сортировка массива..." << endl;

cout.flush();

sort(arr.begin(), arr.end());

cout << "сортировка массива завершена." << endl;

cout.flush();

}

foundBinary = binarySearch(arr, x);

end = chrono::high\_resolution\_clock::now();

auto binary\_time = chrono::duration\_cast<chrono::milliseconds>(end - start);

if (foundBinary)

{

cout << "Число было найдено бинарным поиском за " << binary\_time.count() << " ms" << endl;

}

else

{

cout << "Число не было найдено!!" << endl;

}

// вывод результатов

cout << "Линейный поиск: " << linear\_time.count() << " ms" << endl;

cout << "Двоичный поиск" << (!isSorted ? " с сортировкой данных: " : ": ") << binary\_time.count() << " ms" << endl;

// Вывод результатов в файл

ofstream fout("result.txt");

if (fout.is\_open())

{

fout << "Линейный поиск: " << linear\_time.count() << " ms\n";

fout << "Двоичный поиск" << (!isSorted ? " с сортировкой данных: " : ": ") << binary\_time.count() << " ms\n";

if (!foundLinear)

{

fout << "Линейный поиск не дал результатов и не нашел значение!\n";

}

if (!foundBinary)

{

fout << "Бинарный поиск не дал результатов и не нашел значение!\n";

}

fout.close();

cout << "был создан файл result.txt и в него были сохранены последние пару строк" << endl;

}

else

{

cout << "Ошибка открытия файла!" << endl;

}

return 0;

}