|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет  имени Н. Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н. Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика и системы управления» (ИУ) |

|  |  |
| --- | --- |
| КАФЕДРА | «Информационная безопасность» (ИУ8) |

Лабораторная работа № 1

ПО КУРСУ

«Алгоритмические языки»

на тему «Классы Си++»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-23 |  |  |  | В.С. Ажгирей |
|  | (Группа) |  |  |  | (И. О. Фамилия) |
|  |  |  |  |  |  |
| Преподаватель: |  |  |  |  | М. В. Малахов |
|  |  |  |  |  | (И.О. Фамилия) |

2024

Введение

Цели и задачи работы

Цель работы состоит в изучении основных понятий объектно- ориентированного программирования языка Си++ – классов и объектов, и овладении навыками разработки программ на языке Си++ с использованием объектно-ориентированных средств. Для достижения цели необходимо выполнить следующие задачи:

- изучить необходимые учебные материалы, посвященные основам объектно- ориентированного программирования на языке Си++;

- разработать программу на языке Си++ для решения заданного варианта задания;

- отладить программу;

- выполнить решение контрольного примера с помощью программы и ручной расчет контрольного примера;

- подготовить отчет по лабораторной работе.

Условия для 1 варианта

Класс – сотрудник предприятия. Параметры (поля класса) – ФИО, оклад, надбавка к окладу в %. Статус доступа всех полей private. Класс включает: конструктор, при необходимости функции доступа к полям, функцию печати параметров, функцию вычисления зарплаты (зарплата = оклад + процентная надбавка от оклада\*оклад). Вывести на печать параметры всех сотрудников и суммарную их зарплату.

Основная часть

Файл заголовка sources.hpp:

#pragma once

#include <fstream>

#include <iostream>

#include <string>

#include <typeinfo>

#include <vector>

class EmployeeCompany

{

std::string fullname;

unsigned int salary;

short salary\_supplement;

public:

EmployeeCompany();

explicit EmployeeCompany(std::string, unsigned int, short);

std::string getFullName();

void setFullName(std::string);

unsigned int getSalary();

void setSalary(unsigned int);

short getSalarySupplement();

void setSalarySupplement(short);

float salaryWithAllowance();

void printEmployee();

};

struct TwoValues

{

EmployeeCompany\* dataEmployee;

int quantity = 0;

};

TwoValues readDataDynamic(const std::string&);

std::vector<EmployeeCompany> readDataSTL(const std::string&);

void printData(std::vector<EmployeeCompany>&, const std::string);

void printData(EmployeeCompany\*, const int, const std::string);

Файл описания sources.cpp:

#include "sources.hpp"

EmployeeCompany::EmployeeCompany()

{

this->fullname = "";

this->salary = 0;

this->salary\_supplement = 0;

}

EmployeeCompany::EmployeeCompany(std::string fullname, unsigned int salary, short salary\_supplement)

{

this->fullname = fullname;

this->salary = salary;

this->salary\_supplement = salary\_supplement;

}

std::string EmployeeCompany::getFullName()

{

return fullname;

}

void EmployeeCompany::setFullName(std::string fullname)

{

this->fullname = fullname;

}

unsigned int EmployeeCompany::getSalary()

{

return salary;

}

void EmployeeCompany::setSalary(unsigned int salary)

{

this->salary = salary;

}

short EmployeeCompany::getSalarySupplement()

{

return salary\_supplement;

}

void EmployeeCompany::setSalarySupplement(short salary\_supplement)

{

this->salary\_supplement = salary\_supplement;

}

float EmployeeCompany::salaryWithAllowance()

{

return salary + salary \* salary\_supplement \* 1.f / 100;

}

void EmployeeCompany::printEmployee()

{

std::cout << "Full name: " << fullname << std::endl;

std::cout << "Salary: " << salary << std::endl;

std::cout << "Salary supplement: " << salary\_supplement << std::endl;

}

TwoValues readDataDynamic(const std::string& file\_path)

{

std::ifstream readFile(file\_path);

if (!readFile)

{

std::cerr << "Failed to open file" << std::endl;

return TwoValues();

}

int quantity;

readFile >> quantity;

EmployeeCompany\* dataEmployee = new EmployeeCompany[quantity];

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

{

std::string fullname;

unsigned int salary;

short salary\_supplement;

readFile >> fullname >> salary >> salary\_supplement;

dataEmployee[i] = EmployeeCompany(fullname, salary, salary\_supplement);

}

return TwoValues{ dataEmployee, quantity };

}

std::vector<EmployeeCompany> readDataSTL(const std::string& file\_path)

{

std::ifstream readFile(file\_path);

std::vector<EmployeeCompany> dataEmployee;

if (!readFile)

{

std::cerr << "Failed to open file" << std::endl;

return dataEmployee;

}

int quantity;

readFile >> quantity;

dataEmployee.reserve(quantity);

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

{

std::string fullname;

unsigned int salary;

short salary\_supplement;

readFile >> fullname >> salary >> salary\_supplement;

dataEmployee.push\_back(EmployeeCompany(fullname, salary, salary\_supplement));

}

return dataEmployee;

}

void printData(std::vector<EmployeeCompany>& container, const std::string mode)

{

if (mode == "file")

{

std::string outputFileName;

std::cout << "Enter the name of the output file: ";

std::cin >> outputFileName;

std::ofstream outputFile(outputFileName);

if (!outputFile)

{

std::cerr << "Failed to open file" << std::endl;

return;

}

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

{

outputFile << "Full name: " << container[i].getFullName() << std::endl;

outputFile << "Salary: " << container[i].getSalary() << std::endl;

outputFile << "Salary supplement: " << container[i].getSalarySupplement() << std::endl;

outputFile << "Salary with an allowance: " << container[i].salaryWithAllowance() << std::endl;

outputFile << "#############################################" << std::endl;

}

outputFile.close();

}

else

{

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

{

std::cout << "Full name: " << container[i].getFullName() << std::endl;

std::cout << "Salary: " << container[i].getSalary() << std::endl;

std::cout << "Salary supplement: " << container[i].getSalarySupplement() << std::endl;

std::cout << "Salary with an allowance: " << container[i].salaryWithAllowance() << std::endl;

std::cout << "#############################################" << std::endl;

}

}

std::cout << "Successfully!" << std::endl;

}

void printData(EmployeeCompany\* container, const int quantity, const std::string mode)

{

if (mode == "file")

{

std::string outputFileName;

std::cout << "Enter the name of the output file: ";

std::cin >> outputFileName;

std::ofstream outputFile(outputFileName);

if (!outputFile)

{

std::cerr << "Failed to open file" << std::endl;

return;

}

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

{

outputFile << "Full name: " << container[i].getFullName() << std::endl;

outputFile << "Salary: " << container[i].getSalary() << std::endl;

outputFile << "Salary supplement: " << container[i].getSalarySupplement() << std::endl;

outputFile << "Salary with an allowance: " << container[i].salaryWithAllowance() << std::endl;

outputFile << "#############################################" << std::endl;

}

outputFile.close();

}

else

{

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

{

std::cout << "Full name: " << container[i].getFullName() << std::endl;

std::cout << "Salary: " << container[i].getSalary() << std::endl;

std::cout << "Salary supplement: " << container[i].getSalarySupplement() << std::endl;

std::cout << "Salary with an allowance: " << container[i].salaryWithAllowance() << std::endl;

std::cout << "#############################################" << std::endl;

}

}

delete[] container;

std::cout << "Successfully!" << std::endl;

}

Исполняемый файл main.cpp:

#include "sources.hpp"

int main(int argc, char\* argv[])

{

setlocale(LC\_ALL, "ru");

if (argc < 4)

{

std::cout << "Error: Insufficient command line arguments" << std::endl;

return 1;

}

std::string file\_path = argv[1], container\_type = argv[2], mode = argv[3];

if (container\_type == "stl")

{

std::vector<EmployeeCompany> dataEmployee = readDataSTL(file\_path);

printData(dataEmployee, mode);

}

else

{

TwoValues readData = readDataDynamic(file\_path);

printData(readData.dataEmployee, readData.quantity, mode);

}

return 0;

}

Снимки выполнения работы программы

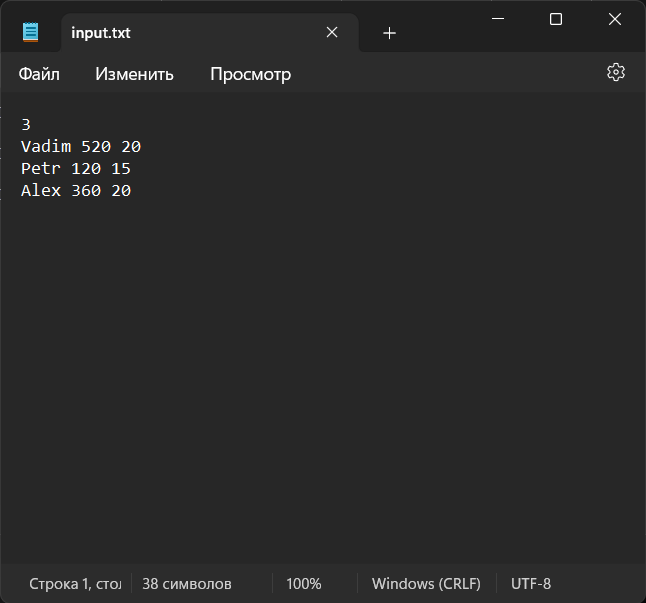


Рисунок 1 – Входные данные

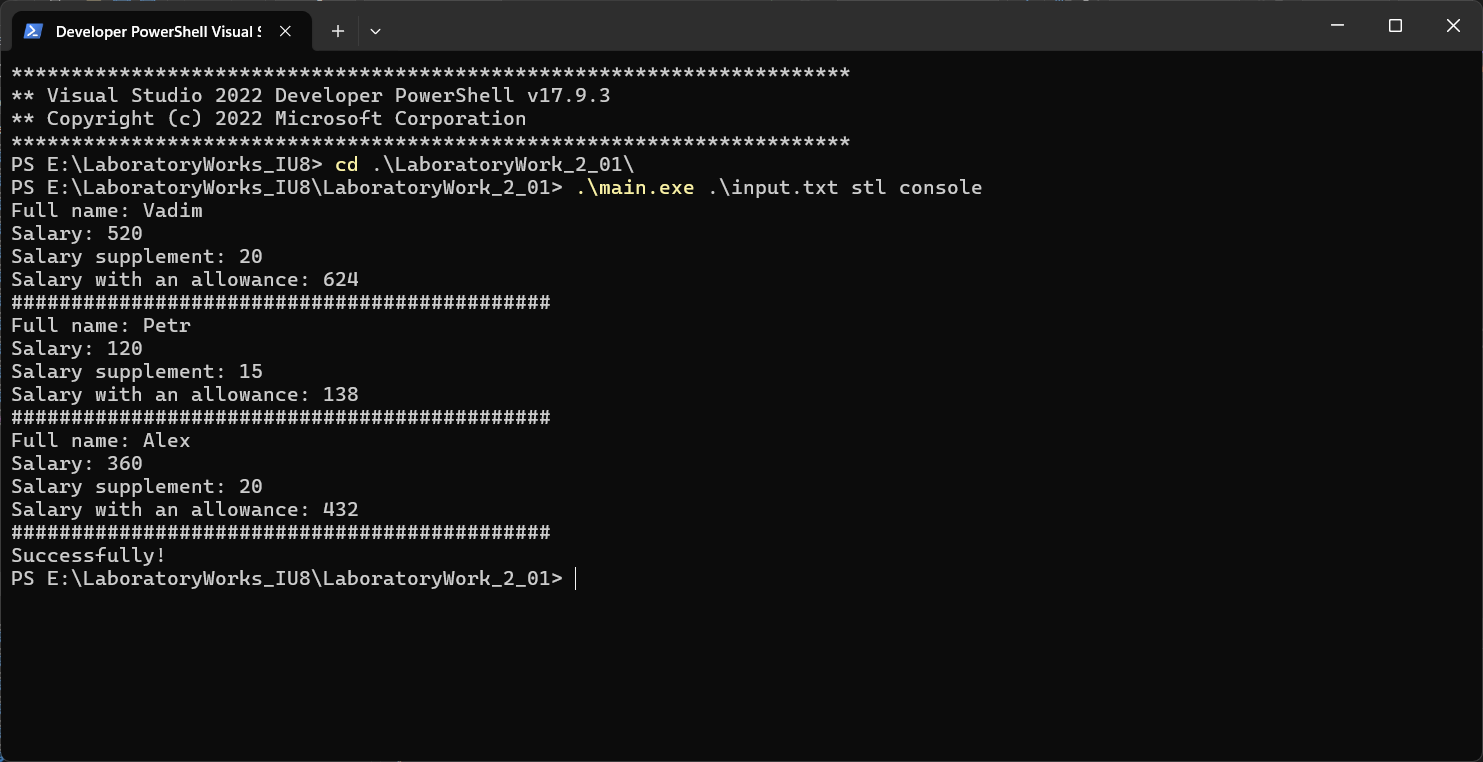


Рисунок 2 – Запуск программы в режиме контейнера std::vector и вывода в консоль

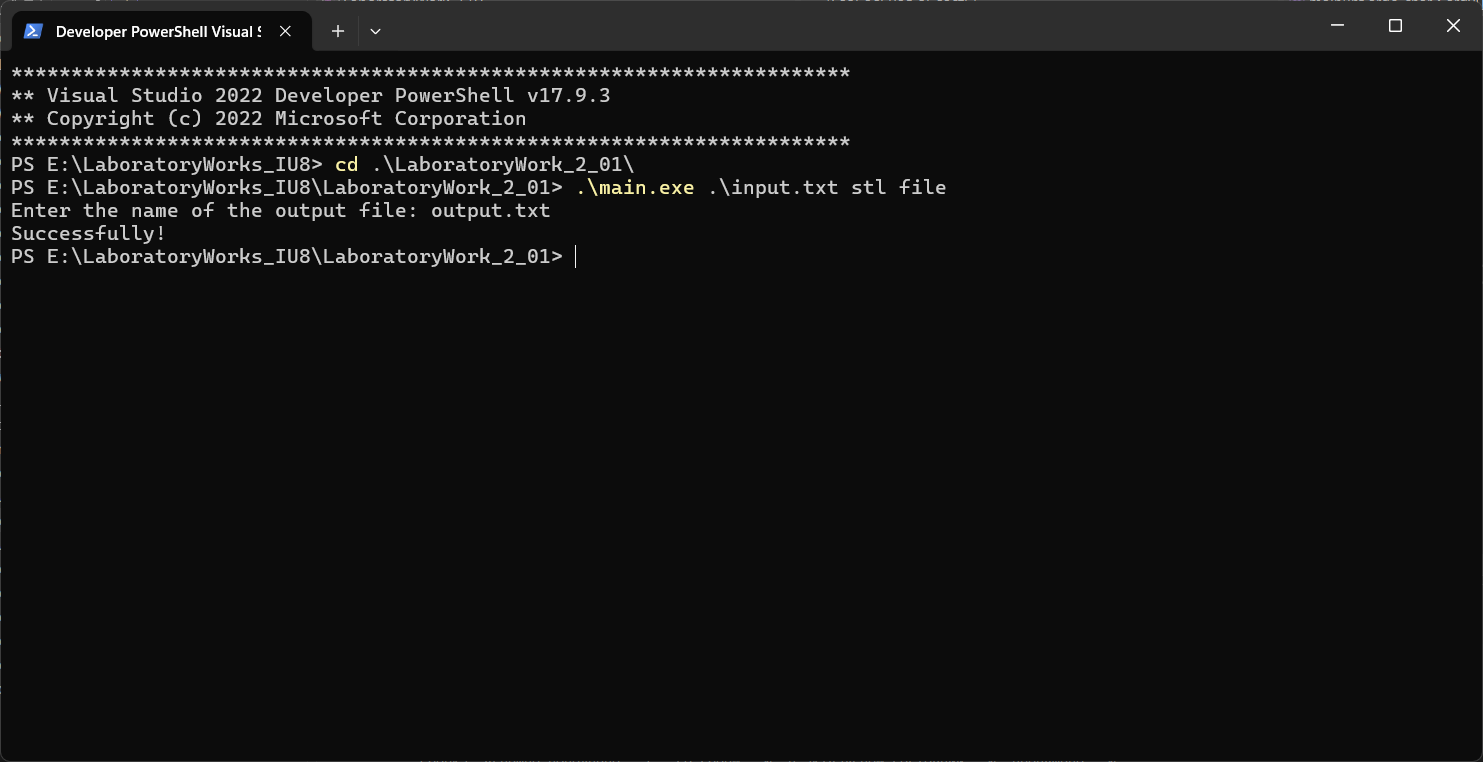


Рисунок 3 – Запуск программы в режиме контейнера std::vector и вывода в файл output.txt

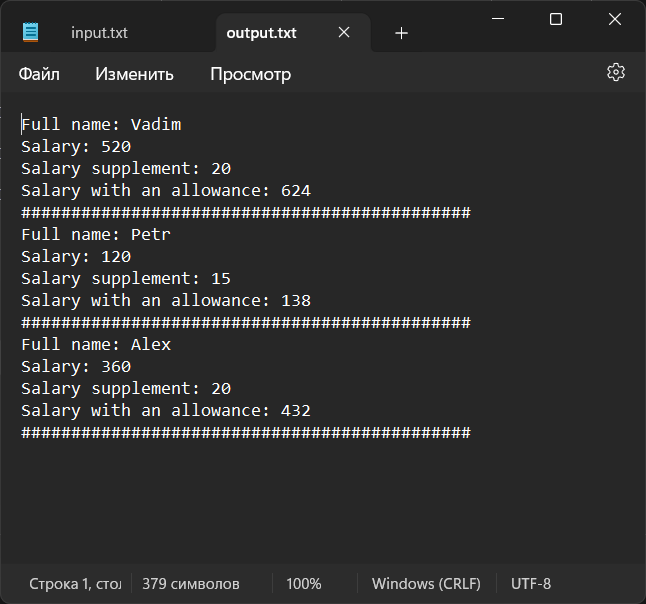


Рисунок 4 – Выходные данные в файле output.txt

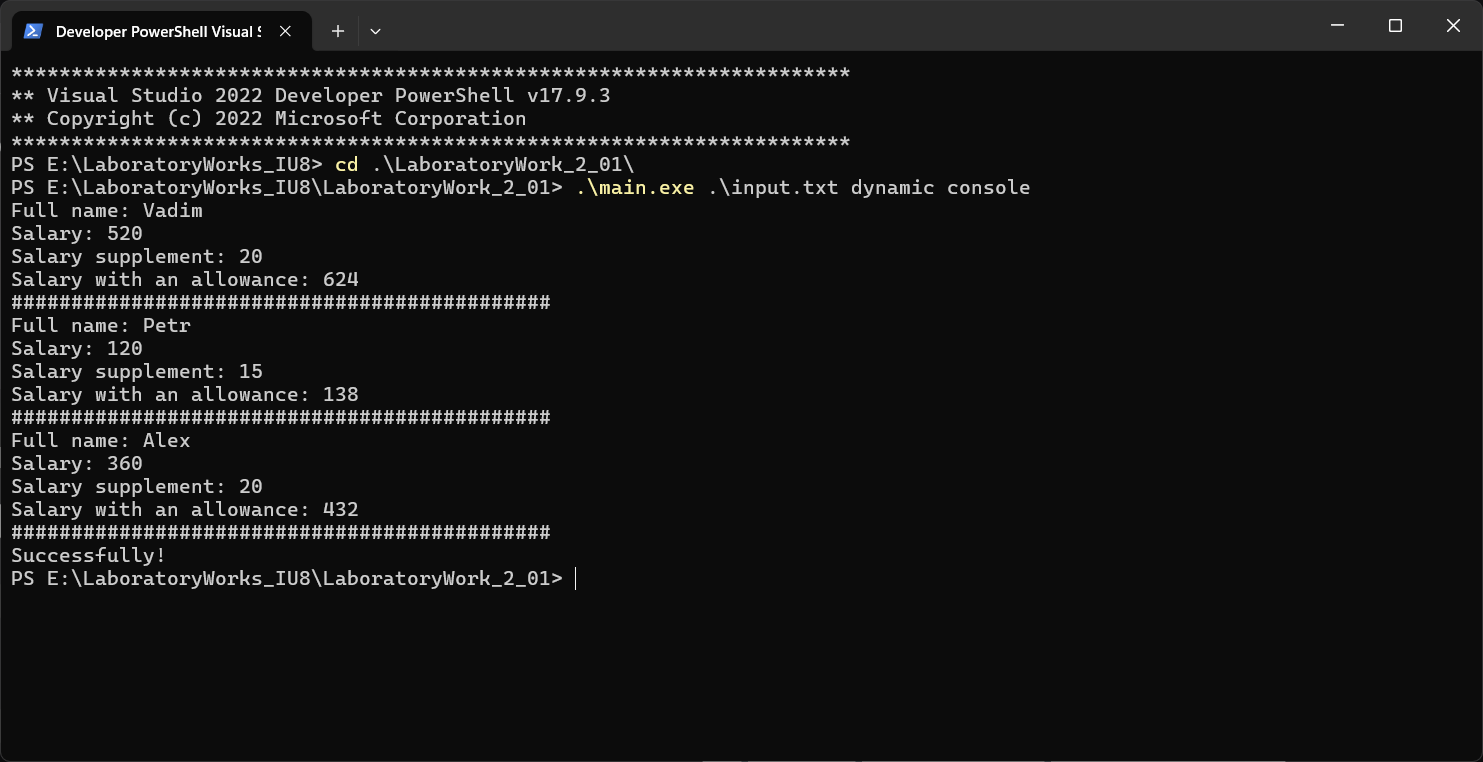


Рисунок 5 – Запуск программы в режиме динамического массива и вывода в консоль

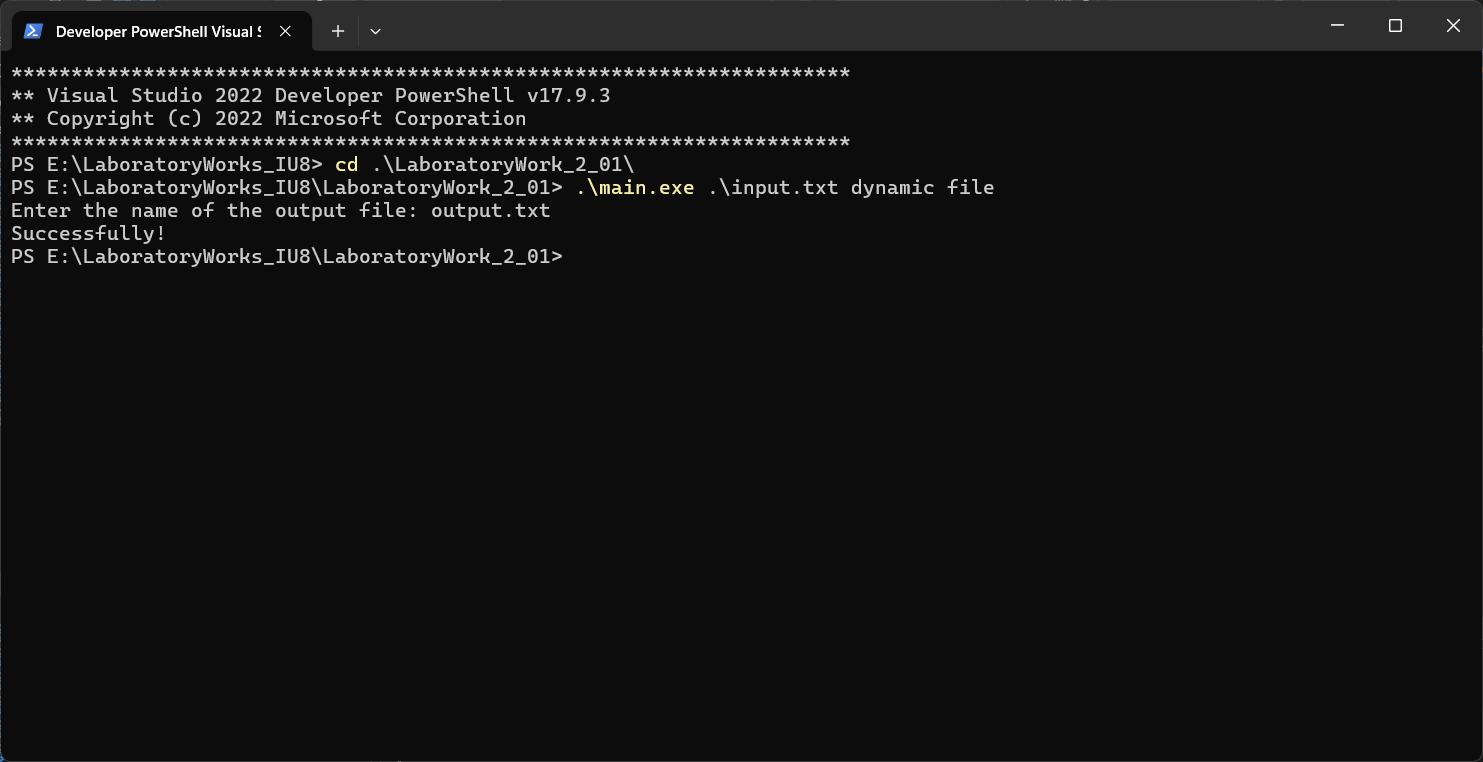


Рисунок 6 – Запуск программы в режиме динамического массива и вывода в файл output.txt

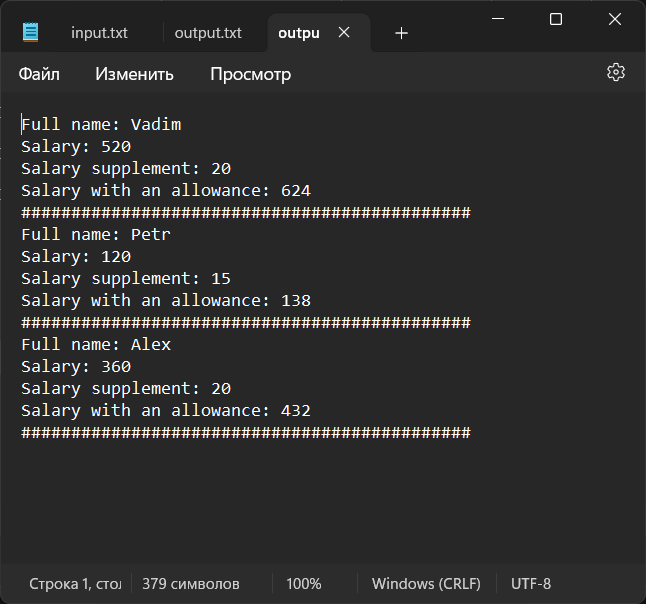


Рисунок 7 – Выходные данные в файле output.txt

Заключение

Задачи лабораторной работы были решены, результаты проверены. Изучены на практике классы в С++. Также изучены основные понятия объектов в С++, такие как конструкторы и деструкторы.