|  |  |
| --- | --- |
|  | **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 № 5

ПО КУРСУ

«Алгоритмические языки»

# на тему «**Изучение использования объектов своих классов в упорядоченных и неупорядоченных контейнерах библиотеки STL (set и map, unordered\_set и unordered\_map)**»

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

2024

Введение

Условия для 4 варианта

Для класса, разработанного в ЛР4, обеспечить возможность добавления объектов в контейнер set (сортировка как указано в задании на ЛР4) и в контейнер unordered\_set. Исходные данные как в ЛР4 читать из файла, вывести на печать для контроля объекты контейнеров.

|  |  |  |
| --- | --- | --- |
| **Параметры приложений** | | **Исходный контейнер list, копируем в vector** |
| Объект- сотрудник (поля: ФИО, дата приема на работу, должность, базовый оклад) | Сортировка по окладу | 4 |
|  |  |

Основная часть

Исходный текст программы:

Employee.h

#include <iostream>

#include <string>

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

class Employee {

protected:

  std::string fio;

  std::string hireDate;

  std::string position;

  int salary;

public:

  Employee();

  Employee(std::string, std::string, std::string, int);

  Employee(const Employee&);

  std::string getFio() const;

  std::string getHireDate() const;

  std::string getPosition() const ;

  int getSalary() const;

*// .Employee(Employee&);*

  void updateSalary(int);

  Employee& operator = (Employee&);

};

class MyHashFunction {

public:

  size\_t operator()(const Employee&) const;

};

bool operator < (const Employee& first, const Employee& second);

*// bool operator <= (const Employee& first, const Employee& second);*

*// bool operator > (const Employee& first, const Employee& second);*

*// bool operator >= (const Employee& first, const Employee& second);*

bool operator == (const Employee& first, const Employee& second);

*// bool operator != (const Employee& first, const Employee& second);*

std::ostream& operator << (std::ostream&, const Employee&);

#endif

Employee.cpp

#include "Employee.h"

#include <iostream>

#include <string>

Employee::Employee() {

  hireDate = position = "undefined";

  salary = 0;

}

Employee::Employee(std::string fio, std::string hireDate, std::string position, int salary) {

*this*->fio = fio;

*this*->hireDate = hireDate;

*this*->position = position;

*this*->salary = salary;

}

Employee::Employee(const Employee& employee) {

  fio = employee.fio;

  hireDate = employee.hireDate;

  position = employee.position;

  salary = employee.salary;

*/\*this->fio = employee.getFio();*

*this->hireDate = employee.getHireDate();*

*this->position = employee.getPosition();*

*this->salary = employee.getSalary();\*/*

}

std::string Employee::getFio() const {

  return fio;

}

std::string Employee::getHireDate() const {

  return hireDate;

}

std::string Employee::getPosition() const {

  return position;

}

int Employee::getSalary() const {

  return salary;

}

void Employee::updateSalary(int salary) {

*this*->salary = salary;

}

Employee& Employee::operator = (Employee& employee) {

*this*->hireDate = employee.getHireDate();

*this*->position = employee.getPosition();

*this*->salary = employee.getSalary();

  return \**this*;

}

*/\*size\_t Employee::HashFunction::operator()(Employee employee) const {*

*return std::hash<int>()(employee.getSalary());*

*}\*/*

size\_t MyHashFunction::operator()(const Employee& employee) const {

  size\_t hash = employee.getSalary();

  for (char sym : employee.getFio()) {

    hash += (int)sym;

  }

  return hash;

}

bool operator < (const Employee& first, const Employee& second) {

  if (first.getSalary() == second.getSalary()) return first.getFio() < second.getFio();

  return first.getSalary() < second.getSalary();

}

*// bool operator > (const Employee& first, const Employee& second) {*

*//   if (first.getSalary() == second.getSalary()) return first.getFio() > second.getFio();*

*//   return first.getSalary() > second.getSalary();*

*// }*

bool operator == (const Employee& first, const Employee& second) {

  return first.getFio() == second.getFio();

}

std::ostream& operator << (std::ostream& out, const Employee& employee) {

  return out << "\tFIO: " << employee.getFio() << std::endl

      << "\thire date: " << employee.getHireDate() << std::endl

      << "\tposition: " << employee.getPosition() << std::endl

      << "\tsalary: " << employee.getSalary() << std::endl;

}

Lab5.cpp

#include "Employee.h"

#include <fstream>

#include <iostream>

#include <set>

#include <string>

#include <unordered\_set>

*// g++ source/lab5/lab5.cpp source/lab5/Employee.cpp -o build/lab5*

std::string tmp;

int main() {

  std::set<Employee> employees\_set;

  std::unordered\_set<Employee, MyHashFunction> employees\_uset;

  int employees\_count;

  std::ifstream fin;

  fin.open("./input/input\_lab5.txt");

  if (!fin.is\_open()) {

    std::cerr << "Error: Could not open input file" << std::endl;

    return 1;

  }

  getline(fin, tmp);

  try {

    employees\_count = std::stoi(tmp);

  } catch (const std::invalid\_argument &e) {

    std::cerr << "Error: Invalid employees' count" << std::endl;

    std::cerr << e.what() << std::endl;

    fin.close();

    return 1;

  }

  std::string fio;

  std::string hireDate;

  std::string position;

  int salary;

  for (int i = 0; i < employees\_count; ++i) {

    getline(fin, fio);

    getline(fin, hireDate);

    getline(fin, position);

    getline(fin, tmp);

    try {

      salary = std::stoi(tmp);

    } catch (const std::invalid\_argument &e) {

      std::cerr << "Error: Invalid salary" << std::endl;

      std::cerr << e.what() << std::endl;

      fin.close();

      return 1;

    }

    employees\_set.insert(Employee(fio, hireDate, position, salary));

    employees\_uset.insert(Employee(fio, hireDate, position, salary));

  }

  fin.close();

  std::ofstream fout\_set;

  fout\_set.open("./output/out\_set\_lab5.txt");

  if (!fout\_set.is\_open()) {

    std::cerr << "Error: Could not open output file" << std::endl;

    return 1;

  }

  for (const auto& employee : employees\_set) {

    fout\_set << employee;

*// std::cout << employee;*

  }

  fout\_set.close();

  std::ofstream fout\_uset;

  fout\_uset.open("./output/out\_uset\_lab5.txt");

  if (!fout\_uset.is\_open()) {

    std::cerr << "Error: Could not open output file" << std::endl;

    return 1;

  }

  for (const auto& employee : employees\_uset) {

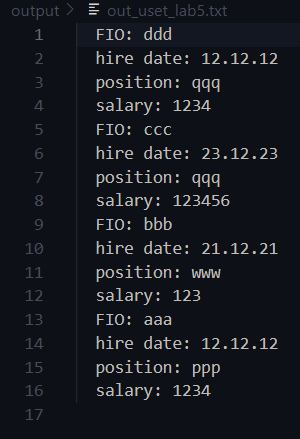
    fout\_uset << employee;

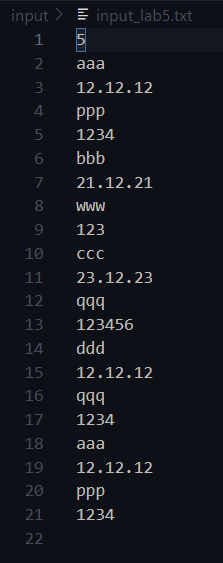
  }

  fout\_uset.close();

  return 0;

}

**Снимки выполнения работы программы

**

Заключение

Задачи лабораторной работы были решены, результаты проверены. Изучены на практике основные понятия объектно-ориентированного программирования языка Си++ – классов и объектов и использования их вместе с контейнерами из библиотеки STL, и приобретены навыки разработки программ на языке Си++ с использованием объектно- ориентированных средств.