|  |  |
| --- | --- |
|  | **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 «**Изучение классов языка Си++**»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-23 |  |  |  | Д.С. Афанасьев |
|  | (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 4 варианта

В работе необходимо разобраться с понятием класса некоторой предметной области и соответствующему ему класса как типа языка Си++, введенного пользователем. Также необходимо знать отличие классов от объектов и назначение и порядок использования основных элементов класса в Си++: полей, методов (функций класса), конструкторов, деструктора. Обратить внимание на способы создания массива объектов класса динамически.

Студент разрабатывает программу на языке Си++ в виде консольного приложения, *в программе необходимо создать массив и последовательной контейнер STL* (vector, deque или list) объектов некоторого класса, данные об объектах читаются из текстового файла. Результаты работы программы и результаты ручного расчета (при необходимости) представляет преподавателю в отчете.

*Входные данные* должны читаться из файла.

Также *разрабатываемое ПО должно состоять* из трёх файлов – файла с объявлением класса (.hpp файл), файла с определением класса (.cpp файл) и файла с функцией main.

Во всех вариантах также необходимо:

Описать класс, включающий заданные поля и методы (функции). Разработать две программы, одна из которых создает массив объектов, а другая последовательный контейнер STL (vector, deque или list выбрать самостоятельно) объектов и выполняет требуемые действия. Все исходные данные для работы программы читаются из текстового файла, созданного в простом редакторе типа «Блокнот». Выходные данные выводятся на консоль и в текстовый файл, для этого использовать одну функцию для вывода.

Класс – аппаратно- программное средство защиты (СЗ) от несанкционированного доступа (НСД). Параметры (поля класса) – название и номер класса защищенности от НСД (Существует семь классов защищенности от НСД, наивысший 1-ый, самый низкий 7, например, если требуется обеспечить защищенность по 3- му классу, то можно использовать СЗ с классами 1, 2 или 3). Статус доступа всех полей private. Класс включает: конструктор, при необходимости функции доступа к полям, функцию, проверяющую можно ли это СЗ использовать для заданного класса (номер заданного класса – параметр функции), функцию печати параметров СЗ. Вывести на печать параметры тех СЗ, которые можно использовать для заданного класса защищенности, номер класса защищенности вводится с клавиатуры.

Основная часть

Исходный текст программы:

*Hardware\_and\_software\_protection.h*

#include <iostream>

#include <string>

#ifndef HARDWARE\_AND\_SOFTWARE\_PROTECTION\_H

#define HSPHARDWARE\_AND\_SOFTWARE\_PROTECTION\_H

class HSP {

private:

  std::string name;

  int protection\_class\_number;

public:

  HSP();

  HSP(const std::string& name, int protection\_class\_number);

  int get\_protection\_class\_number();

  bool can\_be\_used(int protection\_class\_number);

  void print\_info(std::ostream&);

};

#endif

*Hardware\_and\_software\_protection.cpp*

#include "Hardware\_and\_software\_protection.h"

#include <iostream>

#include <string>

HSP::HSP() {

  name = "undefined";

  protection\_class\_number = 0;

  std::cout << "Hello\n";

}

HSP::HSP(const std::string& name, int protection\_class\_number) {

  this->name = name;

  this->protection\_class\_number = protection\_class\_number;

}

int HSP::get\_protection\_class\_number() {

  return protection\_class\_number;

}

bool HSP::can\_be\_used(int protection\_class\_number) {

  if (this->protection\_class\_number == 0) {

    std::cout << "SP is undefined\n";

    return false;

  }

  return this->protection\_class\_number <= protection\_class\_number;

}

void HSP::print\_info(std::ostream &out) {

  out << "\tName: " << name << std::endl;

  out << "\tProtection class number: " << protection\_class\_number << std::endl;

}

*lab1.cpp*

#include "Hardware\_and\_software\_protection.h"

#include<fstream>

#include<iostream>

#include<string>

#include<vector>

std::string tmp;

int solve\_with\_indicator() {

  std::ifstream fin;

  fin.open("./input/input\_lab1.txt");

  if (!fin.is\_open()) {

    std::cerr << "Error: Cannot open input file" << std::endl;

    return 1;

  }

  int number\_of\_HSP;

  getline(fin, tmp);

  try {

    number\_of\_HSP = stoi(tmp);

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

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

    fin.close();

    return 1;

  }

  HSP\*\* protection\_class = new HSP\*[number\_of\_HSP];

  std::string protection\_class\_name;

  int protection\_class\_number;

  for (int i = 0; i < number\_of\_HSP; ++i) {

    getline(fin, protection\_class\_name);

    getline(fin, tmp);

    try {

      protection\_class\_number = stoi(tmp);

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

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

      fin.close();

      return 1;

    }

    protection\_class[i] = new HSP(protection\_class\_name, protection\_class\_number);

  }

  getline(fin, tmp);

  try {

    protection\_class\_number = stoi(tmp);

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

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

    fin.close();

    return 1;

  }

  fin.close();

  std::ofstream fout;

  fout.open("./output/out\_lab1.txt");

  if (!fout.is\_open()) {

    std::cerr << "Error: Cannot open output file" << std::endl;

    return 1;

  }

  std::cout << "Protection classes that can be used for protection class number "

            << protection\_class\_number << " are:\n";

  fout << "Protection classes that can be used for protection class number "

            << protection\_class\_number << " are:\n";

  for (int i = 0; i < number\_of\_HSP; ++i) {

    if (protection\_class[i]->can\_be\_used(protection\_class\_number)) {

      protection\_class[i]->print\_info(std::cout);

      protection\_class[i]->print\_info(fout);

    }

    delete protection\_class[i];

  }

  fout.close();

  delete[] protection\_class;

  return 0;

}

int solve\_with\_vector() {

  std::ifstream fin;

  fin.open("./input/input\_lab1.txt");

  if (!fin.is\_open()) {

    std::cerr << "Error: Cannot open input file" << std::endl;

    return 1;

  }

  int number\_of\_HSP;

  getline(fin, tmp);

  try {

    number\_of\_HSP = stoi(tmp);

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

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

    fin.close();

    return 1;

  }

  std::vector<HSP\*> protection\_class(number\_of\_HSP);

  std::string protection\_class\_name;

  int protection\_class\_number;

  for (int i = 0; i < number\_of\_HSP; ++i) {

    getline(fin, protection\_class\_name);

    getline(fin, tmp);

    try {

      protection\_class\_number = stoi(tmp);

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

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

      fin.close();

      return 1;

    }

    protection\_class[i] =

              new HSP(protection\_class\_name, protection\_class\_number);

  }

  getline(fin, tmp);

  try {

    protection\_class\_number = stoi(tmp);

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

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

    fin.close();

    return 1;

  }

  fin.close();

  std::ofstream fout;

  fout.open("./output/out\_lab1.txt");

  if (!fout.is\_open()) {

    std::cerr << "Error: Cannot open output file" << std::endl;

    return 1;

  }

  std::cout << "\t\tProtection classes that can be used for:\n";

  fout << "\t\tProtection classes that can be used for:\n";

  for (int i = 0; i < number\_of\_HSP; ++i) {

    if (protection\_class[i]->can\_be\_used(protection\_class\_number)) {

      protection\_class[i]->print\_info(std::cout);

      protection\_class[i]->print\_info(fout);

    }

    delete protection\_class[i];

  }

  fout.close();

  return 0;

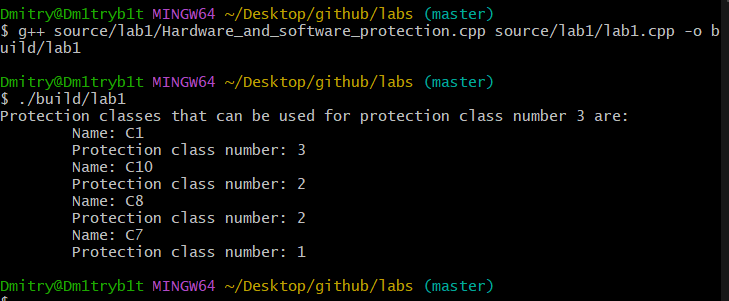
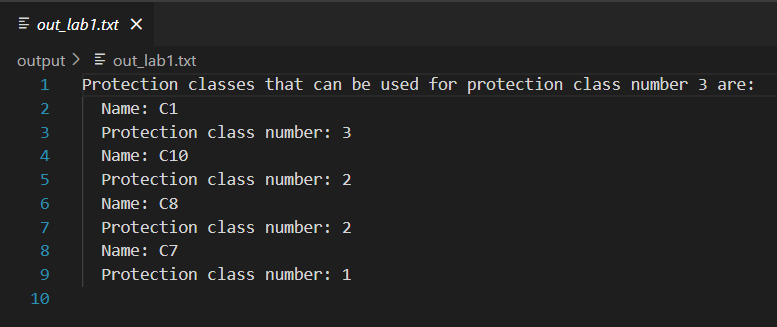
}

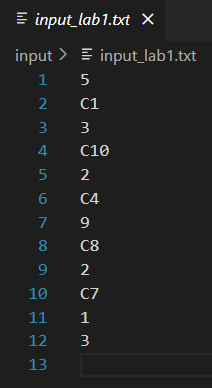
int main() {

  return solve\_with\_indicator();

  // return solve\_with\_vector();

}

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 приобретены навыки разработки программ на языке Си++ с использованием объектно- ориентированных средств.