Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления(АСУ)

НАСЛЕДОВАНИЕ

Отчетпо лабораторной работе №4

по дисциплине «Объектно-ориентированное программирование»

|  |  |
| --- | --- |
|  | Обучающийся гр. \_\_\_442-1\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кадочников А.В.  (подпись) (И.О. Фамилия)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата) |
| **\_\_\_\_\_\_\_\_\_**  (оценка) | Принял  Доцент кафедры АСУ, к.т.н.  (должность, ученая степень, звание)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алфёров С. М.  (подпись) (И.О. Фамилия)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата) |

Томск 2023

Оглавление

[1 Цель работы 3](#_Toc150170317)

[2 Задание 3](#_Toc150170318)

[3 Диаграмма классов 4](#_Toc150170319)

[4 Текст программы 5](#_Toc150170320)

[5 Результат работы 12](#_Toc150170321)

[6Выводы 13](#_Toc150170322)

1 Цель работы

Научиться создавать классы с учетом правил предметной области. Уяснить необходимость скрытия некоторых свойств класса модификаторами private и protected.

2 Задание

Создать класс, имеющий заданные свойства, обеспечить доступ к свойствам (запись и чтение значений свойств) с учетом заданных ограничений. Все конструкторы класса так же должны обеспечивать создание объекта с учетом заданных ограничений. Написать программу, демонстрирующую сохранение ограничений при любых действиях внешней программы над объектом.

Вариант 2

Класс: почтовый адрес

Свойства класса: Улица, дом, квартира

Правила ПО: Первые символы до пробела свойства «улица» указывают на тип. «ул» - улица, «пр» - проспект, «пер» - переулок. На переулке номера домов лежат в диапазоне от 1 до 30, на улице от 1 до 100, на проспекте от 1 до 1000. Если дом частный, то квартира не указывается.

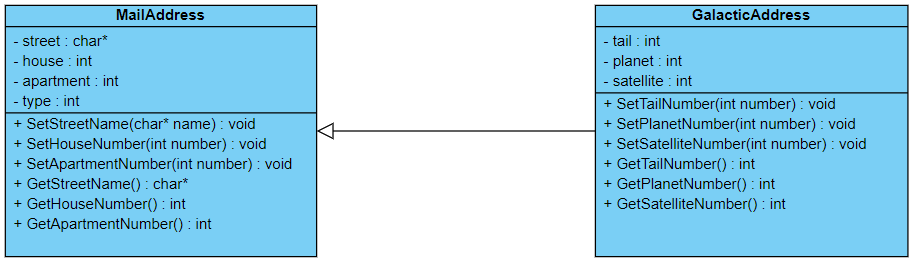
Класс: галактический адрес

Свойства класса: Расстояние от центра галактики, номер хвоста, название звездной системы, номер планеты, номер спутника при необходимости

Правила ПО: Номер хвоста от 1 до 6 Номер планеты от 1 до 10 Номер спутника от 1 до 70

3 Диаграмма классов

Диаграмма классов представлена ниже(Рисунок 1).

Рисунок Диаграмма классов

4 Текст программы

//Вариант 2

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <Windows.h>

#include <string>

#define NO\_VALUE -1

class MailAddress

{

public:

char\* GetStreetName();

int GetHouseNumber();

int GetApartmentNumber();

void SetStreetName(char\* name);

void SetHouseNumber(int number);

void SetApartmentNumber(int number);

private:

char\* street = new char[100] {"ул Ленина"};

int house = 1;

int apartment = 1;

int type = 1;

};

class GalacticAddress: public MailAddress

{

public:

void SetTailNumber(int number);

void SetPlanetNumber(int number);

void SetSatelliteNumber(int number);

int GetTailNumber();

int GetPlanetNumber();

int GetSatelliteNumber();

private:

int tail = 1;

int planet = 1;

int satellite = 1;

};

char\* MailAddress::GetStreetName() {

return this->street;

}

int MailAddress::GetHouseNumber() {

return this->house;

}

int MailAddress::GetApartmentNumber() {

return this->apartment;

}

void MailAddress::SetStreetName(char\* name) {

int fHouse = 0;

char\* temp = new char[100] {};

strcpy(temp, name);

temp = strtok(temp, " ");

int counter = 0;

if ((!strcmp(temp, "ул") || !strcmp(temp, "ул.")) && counter < 1) {

fHouse = 1;

}

else if ((!strcmp(temp, "пр") || !strcmp(temp, "пр.")) && counter < 1) {

fHouse = 2;

}

else if ((!strcmp(temp, "пер") || !strcmp(temp, "пер.")) && counter < 1) {

fHouse = 3;

}

if (fHouse) {

strcpy(street, name);

type = fHouse;

}

else std::cout << "Введено неверное название улицы\n\tбудет установлено значение по умолчанию" << std::endl;

delete[] temp;

}

void MailAddress::SetHouseNumber(int number) {

if (type == 1 && (number > 100 || number < 1)) {

std::cout << "Номер дома на улице не может быть меньше 1 или больше 100\n\tбудет установленно ближайшее значение" << std::endl;

if (number > 100) number = 100;

else number = 1;

}

else if (type == 2 && (number > 1000 || number < 1)) {

std::cout << "Номер дома на проспекте не может быть меньше 1 или больше 1000\n\tбудет установлено ближайшее значение" << std::endl;

if (number > 1000) number = 1000;

else number = 1;

}

else if (type == 3 && (number > 30 || number < 1)) {

std::cout << "Номер дома в переулке не может быть меньше 1 или больше 30\n\tбудет установлено ближайшее значение" << std::endl;

if (number > 30) number = 30;

else number = 1;

}

house = number;

}

void MailAddress::SetApartmentNumber(int number) {

if (number < 1 || number != NO\_VALUE) {

std::cout << "Введено не верное значение квартиры\nБудет установлено значение по умолчанию" << std::endl;

number = 1;

}

apartment = number;

}

void GalacticAddress::SetTailNumber(int number) {

if (number < 1 || number > 6) {

std::cout << "Введено не верное значение хвоста\n\tбудет установлено ближайшее значение" << std::endl;

if (number > 6) number = 6;

else number = 1;

}

tail = number;

}

void GalacticAddress::SetPlanetNumber(int number) {

if (number < 1 || number > 10) {

std::cout << "Введено не верное значение планеты\n\tбудет установлено ближайшее значение" << std::endl;

if (number > 10) number = 10;

else number = 1;

}

planet = number;

}

void GalacticAddress::SetSatelliteNumber(int number) {

if (!((number > 1 && number < 70) || number == -1)){

std::cout << "Введено не верное значение спутника\n\tбудет установлено ближайшее значение" << std::endl;

if (number > 70) number = 70;

else number = 1;

}

satellite = number;

}

int GalacticAddress::GetTailNumber() {

return this->tail;

}

int GalacticAddress::GetPlanetNumber() {

return this->planet;

}

int GalacticAddress::GetSatelliteNumber() {

return this->satellite;

}

void setLocale() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

}

void newMailAddress(MailAddress& Name) {

char\* newStreet = new char[100];

int newHouse = NULL;

int newApartment = NULL;

fseek(stdin, 0, SEEK\_END);

std::cout << "Введите имя улицы: ";

fgets(newStreet, 100, stdin);

newStreet[strcspn(newStreet, "\n")] = 0;

Name.SetStreetName(newStreet);

delete[] newStreet;

std::cout << "Введите номер дома: ";

std::cin >> newHouse;

Name.SetHouseNumber(newHouse);

std::cout << "Этот дом частный? (д/Н): ";

char ch = 0;

std::cin >> ch;

if (ch == 'д')

newApartment = NO\_VALUE;

else{

std::cout << "Введите номер квартиры: ";

std::cin >> newApartment;

}

Name.SetApartmentNumber(newApartment);

}

void outMailAddress(MailAddress& Name) {

std::cout << "Улица: " << Name.GetStreetName() << std::endl;

std::cout << "Дом: " << Name.GetHouseNumber() << std::endl;

if (Name.GetApartmentNumber() != NO\_VALUE) {

std::cout << "Квартира: " << Name.GetApartmentNumber() << std::endl;

}

else {

std::cout << "Частный дом" << std::endl;

}

}

void newGalacticAddress(GalacticAddress& Name) {

int newTail = 0;

int newPlanet = 0;

int newSatellite = 0;

std::cout << "Введите номер хвоста(от 1 до 6)" << std::endl;

std::cin >> newTail;

Name.SetTailNumber(newTail);

std::cout << "Введите номер планеты(от 1 до 10)" << std::endl;

std::cin >> newPlanet;

Name.SetPlanetNumber(newPlanet);

std::cout << "Нужно указать номер спутника?(д/Н): ";

char ch = 0;

std::cin >> ch;

if (ch == 'д') {

std::cout << "Введите номер спутника(от 1 до 70)" << std::endl;

std::cin >> newSatellite;

}

else

newSatellite = NO\_VALUE;

Name.SetSatelliteNumber(newSatellite);

}

void outGalacticAddress(GalacticAddress& Name) {

std::cout << "Хвост: " << Name.GetTailNumber() << std::endl;

std::cout << "Планета: " << Name.GetPlanetNumber() << std::endl;

if(Name.GetSatelliteNumber() != NO\_VALUE)

std::cout << "Спутник: " << Name.GetSatelliteNumber() << std::endl;

}

bool InputAddress(GalacticAddress& Name) {

std::cout << "Указать \033[4mполный\033[24m адресс?(д/Н): ";

char ch = 0;

std::cin >> ch;

bool FullAddress = false;

if (ch == 'д') {

newGalacticAddress(Name);

FullAddress = true;

}

newMailAddress(Name);

return FullAddress;

}

void OutputAddress(GalacticAddress& Name, bool FullAddress) {

if (FullAddress == true) outGalacticAddress(Name);

outMailAddress(Name);

}

int main()

{

setLocale();

GalacticAddress Oleg;

GalacticAddress Anton;

bool FullAddressO = false;

bool FullAddressA = false;

FullAddressO = InputAddress(Oleg);

FullAddressA = InputAddress(Anton);

std::cout << "\n\n" << "Введённые данные" << std::endl;

outGalacticAddress(Oleg);

OutputAddress(Oleg, FullAddressO);

std::cout << "\n" << std::endl;

OutputAddress(Anton, FullAddressA);

return 0;

}

5 Результат работы

Результат работы программы приведён ниже(Рисунок 2 и 3)

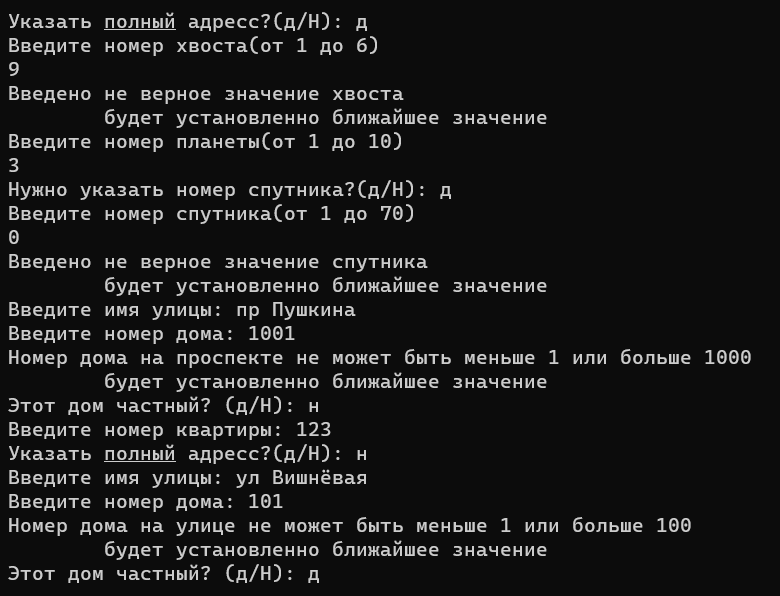
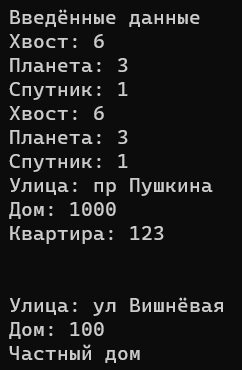


Рисунок 3 Результат работы программы

Рисунок2 Входные данные

6Выводы

В процессе работы над данной лабораторной работой я познакомился с понятием наследования, научился создавать иерархию классов.