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)

ВИРТУАЛЬНЫЕ МЕТОДЫ. МЕХАНИЗМ ПОЗДНЕГО СВЯЗЫВАНИЯ.

Отчетпо лабораторной работе №5

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

Уяснить пользу от виртуальных методов.

2 Задание

Модифицировать программу по лабораторной работе №4. Для каждого класса создать метод вывода на экран всех свойств класса, названия методов должны быть одинаковыми, например show(). В основной программе (функции main) объявить массив указателей на) объявить массив указателей на базовый (родительского) класс. По некоторым указателям массива создать объекты базового (родительского) класса, по остальным указателям массива создать объекты производного (дочернего) класса. С помощью цикла, для каждого указателя массива, вызвать метод show(). Запустить программу и определить не корректность вывода некоторых объектов. Задать метод show() у родительского класса как virtual. Запустить программу и определить отличие работы программы без слова virtual у метода show().

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.

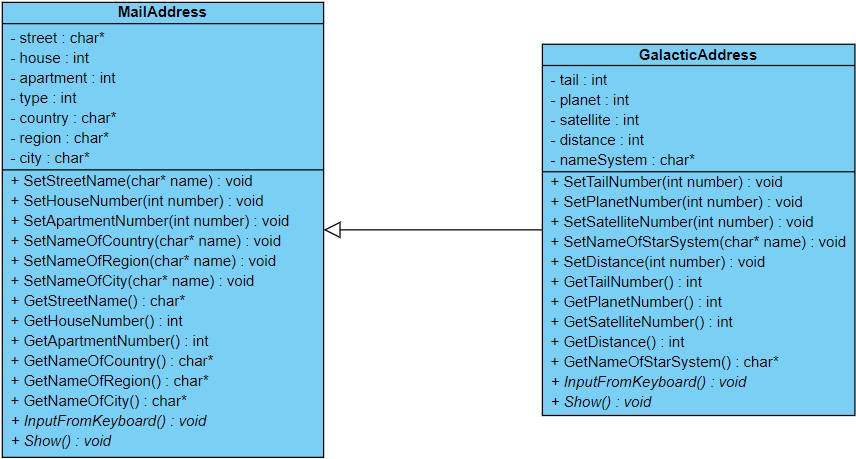
Класс: галактический адрес

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 1 Диаграмма классов

4 Текст программы

//Вариант 2

#include <iostream>

#include <Windows.h>

#include <string>

#define STRINGSIZE 100

#define NO\_VALUE -1

#define STANDART\_INT\_VALUE 1

class MailAddress

{

public:

void SetStreetName(char\* name);

void SetHouseNumber(int number);

void SetApartmentNumber(int number);

void SetNameOfCountry(char\* name);

void SetNameOfRegion(char\* name);

void SetNameOfCity(char\* name);

char\* GetStreetName();

int GetHouseNumber();

int GetApartmentNumber();

char\* GetNameOfCountry();

char\* GetNameOfRegion();

char\* GetNameOfCity();

virtual void InputFromKeyboard();

virtual void Show();

private:

char\* country = new char[STRINGSIZE] {"Россия"};

char\* region = new char[STRINGSIZE] {"Юргинский"};

char\* city = new char[STRINGSIZE] {"Юрга"};

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

int house = STANDART\_INT\_VALUE;

int apartment = STANDART\_INT\_VALUE;

int type = STANDART\_INT\_VALUE;

};

class GalacticAddress : public MailAddress

{

public:

void SetTailNumber(int number);

void SetPlanetNumber(int number);

void SetSatelliteNumber(int number);

void SetDistance(int number);

void SetNameOfStarSystem(char\* name);

int GetTailNumber();

int GetPlanetNumber();

int GetSatelliteNumber();

int GetDistance();

char\* GetNameOfStarSystem();

void InputFromKeyboard();

void Show();

private:

int distance = STANDART\_INT\_VALUE;

char\* nameSystem = new char[STRINGSIZE] {"Солнечная"};

int tail = STANDART\_INT\_VALUE;

int planet = STANDART\_INT\_VALUE;

int satellite = STANDART\_INT\_VALUE;

};

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

int fHouse = 0;

if (name[0] != '\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;

}

delete[] temp;

}

if (fHouse) {

strcpy(street, name);

type = fHouse;

}

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

}

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 установленно 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 установленно 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 установленно значение по умолчанию" << std::endl;

number = 1;

}

apartment = number;

}

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

if (name[0] != '\0') {

strcpy(country, name);

}

else std::cout << "Будет установлено значение по умолчанию" << std::endl;

}

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

if (name[0] != '\0') {

strcpy(region, name);

}

else std::cout << "Будет установлено значение по умолчанию" << std::endl;

}

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

if (name[0] != '\0') {

strcpy(city, name);

}

else std::cout << "Будет установлено значение по умолчанию" << std::endl;

}

char\* MailAddress::GetStreetName() {

return this->street;

}

int MailAddress::GetHouseNumber() {

return this->house;

}

int MailAddress::GetApartmentNumber() {

return this->apartment;

}

char\* MailAddress::GetNameOfCountry(){

return this->country;

}

char\* MailAddress::GetNameOfRegion(){

return this->region;

}

char\* MailAddress::GetNameOfCity(){

return this->city;

}

void MailAddress::InputFromKeyboard() {

char\* newCountry = new char[STRINGSIZE];

char\* newRedion = new char[STRINGSIZE];

char\* newCity = new char[STRINGSIZE];

char\* newStreet = new char[STRINGSIZE];

int newHouse = NULL;

int fHouse = NULL;

int newApartment = NULL;

fseek(stdin, 0, SEEK\_END);

std::cout << "Введите название страны: ";

fgets(newCountry, STRINGSIZE, stdin);

newCountry[strcspn(newCountry, "\n")] = NULL;

SetNameOfCountry(newCountry);

fseek(stdin, 0, SEEK\_END);

std::cout << "Введите название региона: ";

fgets(newRedion, STRINGSIZE, stdin);

newRedion[strcspn(newRedion, "\n")] = NULL;

SetNameOfRegion(newRedion);

fseek(stdin, 0, SEEK\_END);

std::cout << "Введите название города: ";

fgets(newCity, STRINGSIZE, stdin);

newCity[strcspn(newCity, "\n")] = NULL;

SetNameOfCity(newCity);

fseek(stdin, 0, SEEK\_END);

std::cout << "Введите имя улицы: ";

fgets(newStreet, STRINGSIZE, stdin);

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

SetStreetName(newStreet);

delete[] newCountry;

delete[] newRedion;

delete[] newCity;

delete[] newStreet;

std::cout << "Введите номер дома: " << std::endl;

std::cin >> newHouse;

SetHouseNumber(newHouse);

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

char ch = NULL;

std::cin >> ch;

if (ch == 'д')

SetApartmentNumber(NO\_VALUE);

else {

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

std::cin >> newApartment;

SetApartmentNumber(newApartment);

}

}

void MailAddress::Show() {

std::cout << "Страна: " << GetNameOfCountry() << std::endl;

std::cout << "Регион: " << GetNameOfRegion() << std::endl;

std::cout << "Город: " << GetNameOfCity() << std::endl;

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

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

if (GetApartmentNumber() == NO\_VALUE) {

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

}

else {

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

}

}

void GalacticAddress::SetTailNumber(int number) {

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

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

if (number > 70) number = 70;

else number = 1;

}

satellite = number;

}

void GalacticAddress::SetDistance(int number) {

if (number == 0) {

std::cout << "Дистанция не может быть равна нулю\n\tбудет установлено значение по умолчанию\n";

number = STANDART\_INT\_VALUE;

}

distance = number;

}

void GalacticAddress::SetNameOfStarSystem(char\* name) {

if (name[0] != '\0') {

strcpy(nameSystem, name);

}

else std::cout << "Будет установлено значение по умолчанию\n" << std::endl;

}

int GalacticAddress::GetTailNumber() {

return this->tail;

}

int GalacticAddress::GetPlanetNumber() {

return this->planet;

}

int GalacticAddress::GetSatelliteNumber() {

return this->satellite;

}

int GalacticAddress::GetDistance() {

return this->distance;

}

char\* GalacticAddress::GetNameOfStarSystem() {

return this->nameSystem;

}

void GalacticAddress::InputFromKeyboard() {

int newDistance = NULL;

char\* newNameSystem = new char[STRINGSIZE];

int newTail = NULL;

int newPlanet = NULL;

int newSatellite = NULL;

std::cout << "Растояние от центра галактики: " << std::endl;

std::cin >> newDistance;

SetDistance(newDistance);

fseek(stdin, 0, SEEK\_END);

std::cout << "Введите название звёздной системы: ";

fgets(newNameSystem, STRINGSIZE, stdin);

newNameSystem[strcspn(newNameSystem, "\n")] = NULL;

SetNameOfStarSystem(newNameSystem);

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

std::cin >> newTail;

SetTailNumber(newTail);

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

std::cin >> newPlanet;

SetPlanetNumber(newPlanet);

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

char ch = NULL;

std::cin >> ch;

if (ch == 'д') {

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

std::cin >> newSatellite;

if (newSatellite < 1 || newSatellite > 70)

std::cout << "Номер спутника не может быть меньше 1 или больше 70" << std::endl;

else {

SetSatelliteNumber(newSatellite);

}

}

else

SetSatelliteNumber(NO\_VALUE);

MailAddress::InputFromKeyboard();

}

void GalacticAddress::Show() {

std::cout << "Расстояние от центра галактики: " << GetDistance() << std::endl;

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

std::cout << "Название звездной системы: " << GetNameOfStarSystem() << std::endl;

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

if (GetSatelliteNumber() != NO\_VALUE)

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

MailAddress::Show();

}

void setLocale() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

}

int main() {

setLocale();

const int arraySize = 2;

MailAddress\* addresses[arraySize]{};

for (int i = 0; i < arraySize; i++) {

if (i % 2 == 0) {

addresses[i] = new MailAddress();

}

else {

addresses[i] = new GalacticAddress();

}

addresses[i]->InputFromKeyboard();

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

}

//char\* copy = new char[100] {};

//strcpy(copy, "adkjhaf");

//addresses[0]->SetNameOfCountry(copy);

//delete[](copy);

//addresses[1]->SetHouseNumber(101);

for (int i = 0; i < arraySize; i++) {

addresses[i]->Show();

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

}

for (int i = 0; i < arraySize; i++) {

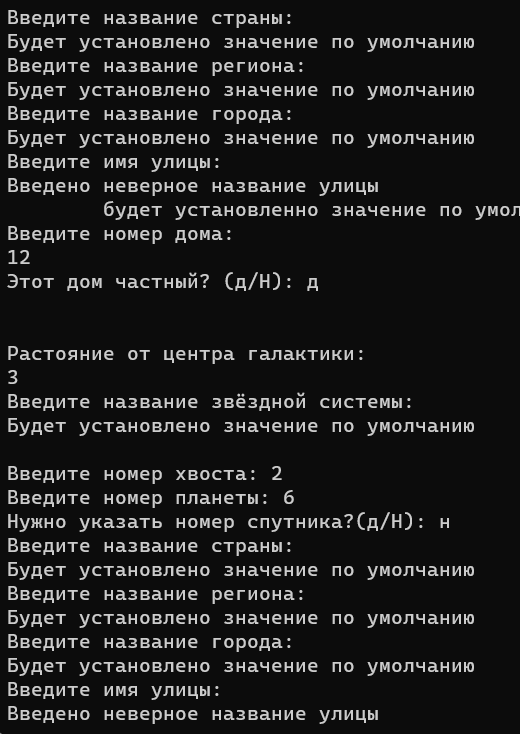
delete addresses[i];

}

return 0;

}

5 Результат работы

Результат работы программы приведён ниже(Рисунок 2 и 3)

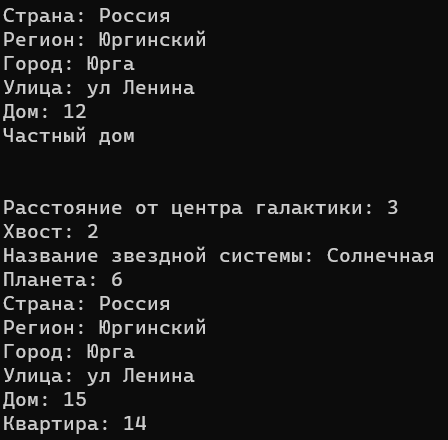


Рисунок 3 Результат работы программы

Рисунок2 Входные данные

6Выводы

В процессе работы над данной лабораторной работой я уяснил пользу от виртуальных методов.