**Курс «Объектно-ориентированное программирование на C++»**

**Встреча №18**

Тема: Наследование

Задания для самостоятельной работы:

**Задание №1.** Создайте класс Student, который будет содержать информацию о студенте.

С помощью механизма наследования, реализуйте класс Aspirant (аспирант - студент, который готовится к защите кандидатской работы) производный от Student.

**Задание №2.** Создайте класс Passport (паспорт), который будет содержать паспортную информацию о гражданине Украины.

С помощью механизма наследования, реализуйте класс ForeignPassport (загран.паспорт) производный от Passport.

Напомним, что заграничный паспорт содержит помимо паспортных данных, также данные о визах, номер заграничного паспорта.

**Задание №3.** Создать абстрактный базовый класс «Транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка».

Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

Завдання 1

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include <iostream>

#include<conio.h>

#include<time.h>

using namespace std;

class Student

{

protected:

char\* PIB;

char\* forma\_navchanna;

char\* profil;

char\* group;

public:

Student()

{

PIB = NULL;

forma\_navchanna = NULL;

profil = NULL;

group = NULL;

}

~Student()

{

if (PIB != NULL)

delete[] PIB;

if (forma\_navchanna != NULL)

delete[] forma\_navchanna;

if (profil != NULL)

delete[] profil;

if (group != NULL)

delete[] group;

}

void Add(const char\* PIB, const char\* forma\_navchanna, const char\* profil, const char\* group)

{

this->PIB = new char[strlen(PIB) + 1];

this->forma\_navchanna = new char[strlen(forma\_navchanna) + 1];

this->profil = new char[strlen(profil) + 1];

this->group = new char[strlen(group) + 1];

strcpy(this->PIB, PIB);

strcpy(this->forma\_navchanna, forma\_navchanna);

strcpy(this->profil, profil);

strcpy(this->group, group);

}

virtual void Show()

{

cout << "-----------------------\n";

cout << PIB << endl;

cout << forma\_navchanna << endl;

cout << profil << endl;

cout << group << endl;

cout << "-----------------------\n";

}

};

class Aspirant: public Student

{

protected:

char\* topic;

public:

Aspirant()

{

topic = NULL;

}

~Aspirant()

{

if (topic != NULL)

delete[] topic;

}

void Add(const char\* PIB, const char\* forma\_navchanna, const char\* profil, const char\* group, const char\* topic)

{

Student::Add(PIB, forma\_navchanna, profil, group);

this->topic = new char[strlen(topic) + 1];

strcpy(this->topic, topic);

}

void Show()

{

cout << "-----------------------\n";

cout << PIB << endl;

cout << forma\_navchanna << endl;

cout << profil << endl;

cout << group << endl;

cout << topic << endl;

cout << "-----------------------\n";

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

Student a;

Aspirant b;

a.Add("Lashch Anton Vadymovich", "Денна", "Автоматизація", "AB-11");

a.Show();

b.Add("Lashch Anton Vadymovich", "Денна", "Автоматизація", "AB-11", "Автоматизація фабрики ляльок");

b.Show();

return 0;

}

Завдання 2

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include <iostream>

#include<conio.h>

#include<time.h>

using namespace std;

class Passport

{

protected:

char\* number;

char\* PIB;

char\* date\_of\_birth;

char\* birthplace;

public:

Passport()

{

number = PIB = date\_of\_birth = birthplace = NULL;

}

~Passport()

{

if (number != NULL)

delete[] number;

if (PIB != NULL)

delete[] PIB;

if (date\_of\_birth != NULL)

delete[] date\_of\_birth;

if (birthplace != NULL)

delete[] birthplace;

}

void Add(const char\* number, const char\* PIB, const char\* date\_of\_birth, const char\* birthplace)

{

this->number = new char[strlen(number) + 1];

this->PIB = new char[strlen(PIB) + 1];

this->date\_of\_birth = new char[strlen(date\_of\_birth) + 1];

this->birthplace = new char[strlen(birthplace) + 1];

strcpy(this->number, number);

strcpy(this->PIB, PIB);

strcpy(this->date\_of\_birth, date\_of\_birth);

strcpy(this->birthplace, birthplace);

}

virtual void Show()

{

cout << "-----------------------------------\n";

cout << number << endl;

cout << PIB << endl;

cout << date\_of\_birth << endl;

cout << birthplace << endl;

cout << "-----------------------------------\n";

}

};

class ForeignPassport:public Passport

{

protected:

char\* visu;

public:

ForeignPassport()

{

visu = NULL;

}

~ForeignPassport()

{

if (visu != NULL)

delete[] visu;

}

void Add(const char\* number, const char\* PIB, const char\* date\_of\_birth, const char\* birthplace, const char\* visu)

{

Passport :: Add(number, PIB, date\_of\_birth, birthplace);

this->visu = new char[strlen(visu) + 1];

strcpy(this->visu, visu);

}

void Show()

{

cout << "-----------------------------------\n";

cout << number << endl;

cout << PIB << endl;

cout << date\_of\_birth << endl;

cout << birthplace << endl;

cout << visu << endl;

cout << "-----------------------------------\n";

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

Passport a;

ForeignPassport b;

a.Add("A3fds434g", "Derkach Vasil Ivanovich", "17.04.1976", "Lutsk");

a.Show();

b.Add("A3fds434g", "Derkach Vasil Ivanovich", "17.04.1976", "Lutsk", "Єгипет, Лондон");

b.Show();

return 0;

}

Завдання 3

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include <iostream>

#include<conio.h>

#include<time.h>

using namespace std;

class Transport

{

protected:

char\* name;

int number\_of\_wheels;

int places\_to\_sit;

int time\_for\_transportaion\_for\_10km;

float cost\_for\_passengers\_for\_10km;

float cost\_for\_load\_for\_10km;

public:

Transport()

{

name = NULL;

}

~Transport()

{

if (name != NULL)

delete[] name;

}

void Add(const char\* name, int number\_of\_wheels, int places\_to\_sit, int time\_for\_transportaion\_for\_10km, float cost\_for\_passengers\_for\_10km, float cost\_for\_load\_for\_10km)

{

this->name = new char[strlen(name) + 1];

strcpy(this->name, name);

this->number\_of\_wheels = number\_of\_wheels;

this->places\_to\_sit = places\_to\_sit;

this->time\_for\_transportaion\_for\_10km = time\_for\_transportaion\_for\_10km;

this->cost\_for\_passengers\_for\_10km = cost\_for\_passengers\_for\_10km;

this->cost\_for\_load\_for\_10km = cost\_for\_load\_for\_10km;

}

void Show()

{

cout << "------------------------\n";

cout << name << endl;

cout << "Кількість колес - " << number\_of\_wheels << endl;

cout << "Сидячих місць - " << places\_to\_sit<<endl;

cout << "Час для проїзду 10км - " << time\_for\_transportaion\_for\_10km << "хв"<<endl;

cout << "Ціна проїзду 10км для пасажира - " << cost\_for\_passengers\_for\_10km <<"грн"<< endl;

cout << "Ціна перевезення 10км грузу - " << cost\_for\_load\_for\_10km << "грн" << endl;

}

};

class Car :public Transport

{

protected:

char\* engine\_characteristics;

char\* fuel;

public:

Car()

{

engine\_characteristics = fuel = NULL;

}

~Car()

{

if (engine\_characteristics != NULL)

delete[] engine\_characteristics;

if (fuel != NULL)

delete[] fuel;

}

void Add(const char\* name, int number\_of\_wheels, int places\_to\_sit, int time\_for\_transportaion\_for\_10km, float cost\_for\_passengers\_for\_10km, float cost\_for\_load\_for\_10km, const char\* engine\_characteristics, const char\* fuel)

{

Transport::Add(name, number\_of\_wheels, places\_to\_sit, time\_for\_transportaion\_for\_10km, cost\_for\_passengers\_for\_10km, cost\_for\_load\_for\_10km);

this->engine\_characteristics = new char[strlen(engine\_characteristics) + 1];

strcpy(this->engine\_characteristics, engine\_characteristics);

this->fuel = new char[strlen(fuel) + 1];

strcpy(this->fuel, fuel);

}

void Show()

{

Transport::Show();

cout << "Двигун - " << this->engine\_characteristics << endl;

cout << "Пальне - " << this->fuel << endl;

}

};

class Bike :public Transport

{

protected:

char\* type\_of\_bicycle;

public:

Bike()

{

type\_of\_bicycle = NULL;

}

~Bike()

{

if (type\_of\_bicycle != NULL)

delete[] type\_of\_bicycle;

}

void Add(const char\* name, int number\_of\_wheels, int places\_to\_sit, int time\_for\_transportaion\_for\_10km, float cost\_for\_passengers\_for\_10km, float cost\_for\_load\_for\_10km, const char\* type\_of\_bicycle)

{

Transport::Add(name, number\_of\_wheels, places\_to\_sit, time\_for\_transportaion\_for\_10km, cost\_for\_passengers\_for\_10km, cost\_for\_load\_for\_10km);

this->type\_of\_bicycle = new char[strlen(type\_of\_bicycle) + 1];

strcpy(this->type\_of\_bicycle, type\_of\_bicycle);

}

void Show()

{

Transport::Show();

cout << "Тип велосипеда - " << type\_of\_bicycle << endl;

}

};

class Wagon :public Transport

{

protected:

int number\_of\_horses;

public:

Wagon()

{

}

~Wagon()

{

}

void Add(const char\* name, int number\_of\_wheels, int places\_to\_sit, int time\_for\_transportaion\_for\_10km, float cost\_for\_passengers\_for\_10km, float cost\_for\_load\_for\_10km, int number\_of\_horses)

{

Transport::Add(name, number\_of\_wheels, places\_to\_sit, time\_for\_transportaion\_for\_10km, cost\_for\_passengers\_for\_10km, cost\_for\_load\_for\_10km);

this->number\_of\_horses = number\_of\_horses;

}

void Show()

{

Transport::Show();

cout << "Кількість коней для запрягання - " << number\_of\_horses << endl;

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

Car a;

Bike b;

Wagon c;

a.Add("BMW", 4, 5, 4, 200.5, 50, "V8", "A-95");

a.Show();

b.Add("Rover", 2, 2, 20, 50, 20, "Спортивний");

b.Show();

c.Add("Віз 2007", 4, 6, 60, 100, 30, 1);

c.Show();

return 0;

}