**Лабораторна робота №5**

**Тема:** Віртуальні функції та поліморфізм.

**Мета:** Практично ознайомитись з поняттям поліморфізму, його застосуванням та вивчити механізм його реалізації за допомогою віртуальних функцій.

***ПОРЯДОК ВИКОНАННЯ РОБОТИ***

1. Ознайомитись з теоретичним матеріалом.

2. Проаналізувати результати роботи програми у наведеному прикладі.

3. Напишіть програму згідно завдання 1.

**Завдання 1**. Нехай є видавнича компанія, яка описана в завданні 1 попередньої лабораторної роботи, яка продає і книги, і аудіо версії друкованої продукції. Як і в тому завданні, створіть клас publication, який зберігає назву (фактично, рядок) і ціну (типу float) публікаціі. Створіть два похідних класа: book, в якому відбувається зміна лічильника сторінок (типа int), і tape, в якому відбувається зміна лічильника записаних на касету хвилин. Кожен з класів повинен мати метод getdata(), який буде запитувати інформацію у користувача, і putdata() для виведення даних на екран. Напишіть main(), де створювався б масив покажчиків на клас publication. У циклі запитуйте у користувача дані про конкретні книги або касети, використовуйте new для створення нового об'єкта book або tape. Зіставляйте покажчик в масиві з об'єктом. Коли користувач закінчить введення вихідних даних, виведіть результат для всіх введених книг і касет, використовуючи цикл for і єдиний вираз: pubаrr[i]->putdata(); для виведення даних про кожен об'єкт з масиву.

4. Напишіть програму згідно завдання 2.

**Завдання 2**. Взявши за основу програму із завдання 1, додайте метод типу bool, який називається isOveersize (), до класів book і tape. Припустимо, книга, в якій більше 800 сторінок, або касета з часом програвання якої більше 90

хвилин, будуть вважатися об'єктами з перевищенням розміру. До цієї функції можна звертатися з main(), а результат її роботи виводити у вигляді рядка «Перевищення розміру!» Для відповідних книг і касет. Припустимо, об'єкти класів book і tape повинні бути доступні через покажчики на них, що зберігаються в масиві типу publication.

5\*. Для заданого варіанта індивідуального завдання написати програму з використанням динамічного поліморфізму.

**Код програми:**

#include <string>

#include <iostream>

using namespace std;

class Publication

{

string name;

float price;

public:

Publication(){}

virtual void *getData*()

{

cout <<"Name : "<< name << endl << "Price: " << price << endl;

}

virtual void *putData*() {cout << "Write name: "; cin >> name;

cout << "Write price: "; cin >> price; }

virtual bool *isOversized*(){return false;}

};

class Book: public Publication

{

int pageAmount;

public:

Book(){}

void *getData*() override{cout << "Book: " << endl; Publication::getData(); cout << "Page amount: " << pageAmount << endl; }

void *putData*() override{Publication::putData(); cout << "Write page amount: "; cin >> pageAmount; }

bool *isOversized*() override{ if(pageAmount > 800) return true; else return false; }

};

class Tape: public Publication

{

int writeTime;

public:

Tape(){}

void *getData*() override{cout << "Tape: " << endl; Publication::getData(); cout << "Write time: " << writeTime << endl; }

void *putData*() override{Publication::putData(); cout << "Write writing time: "; cin >> writeTime; }

bool *isOversized*() override{ if(writeTime > 90) return true; else return false; }

};

int main()

{

Publication \*\*pubarr;

cout << "Write count: ";

int c;

cin >> c;

pubarr = new Publication\*[c];

for(int i = 0; i < c; i++)

{

cout << "Choose publication(Book(1), Tape(2)): ";

int type = 1;

cin >> type;

if(type == 2)

pubarr[i] = new Tape();

else

pubarr[i] = new Book();

pubarr[i]->*putData*();

}

for(int i = 0; i < c; i++)

{

pubarr[i]->*getData*();

if(pubarr[i]->*isOversized*()) cout << "Oversized!" << endl;

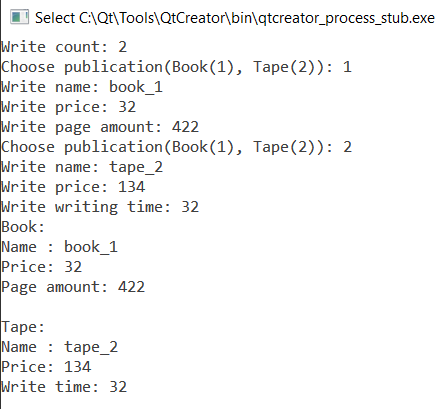
cout << endl;

}

return 0;

}

**Вивід програми:**

****

**Індивідуальне завдання:**

Створити клас ЧОТИРИКУТНИК з полями – координатами вершин. Визначити віртуальну функцію обчислення площі. Створити похідні класи ТРАПЕЦІЯ, РОМБ зі своїми функціями обчислення площі. Для перевірки

**Код програми:**

#include <iostream>

using namespace std;

class Quandrandl

{

protected:

int a,b,c,d;

public:

Quandrandl(){a=0;b=0;c=0;d=0;}

~Quandrandl(){}

virtual void *putDate*()

{

cout<<"Write a,b,c,d"<<endl;

cin>>a>>b>>c>>d;

}

virtual void *S*()

{

cout<<"S = "<<a\*b\*c\*d<<endl;

}

};

class Trapeze:public Quandrandl

{

int h;

public:

Trapeze(int \_a, int \_b,int \_h){a=\_a; b=\_b; h=\_h;}

Trapeze(){}

~Trapeze(){}

void *S*()

{

int S;

S=(a+b)/2\*h;

cout<<"S = "<<S<<endl;

}

void *putDate*()

{

cout<<"Write a,b,h"<<endl;

cin>>a>>b>>h;

}

};

class Romb:public Quandrandl

{

int h;

public:

Romb(int \_a,int \_h){a=\_a; h=\_h;}

Romb(){}

~Romb(){}

void *S*()

{

int S;

S=a\*h;

cout<<"S = "<<S<<endl;

}

void *putDate*()

{

cout<<"Write a,h"<<endl;

cin>>a>>h;

}

};

int main()

{

Trapeze obj\_1(3,2,4 );

Romb obj\_3(3,5);

////////////////////

Quandrandl\* mas[2];

////////////////////

mas[0]=&obj\_1;

mas[1]=&obj\_3;

for(int i=0;i<2;i++)

{

mas[i]->*S*();

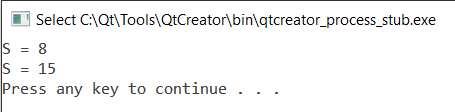
}

system("pause");

return 0;

}

**Вивід програми:**



**Висновок :** практично ознайомився з поняттям поліморфізму, його застосуванням та вивчив механізм його реалізації за допомогою віртуальних функцій.