***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***

**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.

Код програми:

#include <QCoreApplication>

#include <string.h>

#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(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

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;

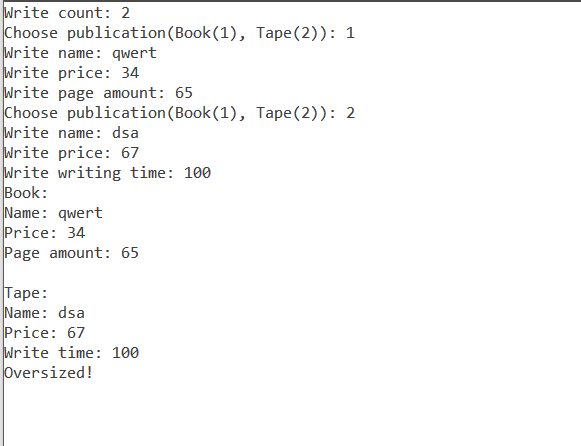
cout << endl;

}

return a.exec();

}

Вивід програми:



**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.

Код програми:

#include <QCoreApplication>

#include <string.h>

#include <iostream>

using namespace std;

class Point

{

float x,y;

public:

Point(float \_x, float \_y){x = \_x; y = \_y;}

Point():Point(0,0){}

float GetX(){return x;}

float GetY(){return y;}

void SetX(float \_x){ x = \_x;}

void SetY(float \_y){ x = \_y;}

string Get(){return "(" + to\_string(x) + ";" + to\_string(y) + ")";}

float GetLength(){return (float)sqrt(pow(x,2)+pow(y,2));}

float GetAngle(){return atan2(y,x);}

float DistanceBetween(Point p)

{

return (float)sqrt(pow(x-p.x,2)+pow(y-p.y,2));

}

Point& operator-(Point& p)

{

return \*new Point(x-p.x, y-p.y);

}

};

class Fourangle

{

Point verticles[4];

public:

Fourangle():Fourangle(\*new Point(0,0), \*new Point(0,0), \*new Point(0,0), \*new Point(0,0)){}

Fourangle(Point &v1, Point &v2, Point &v3, Point &v4)

{

verticles[0] = v1;

verticles[1] = v2;

verticles[2] = v3;

verticles[3] = v4;

}

~Fourangle() { }

void SetVerticle(int pos, Point &p)

{

verticles[pos] = p;

}

Point& GetVerticle(int pos)

{

return verticles[pos];

}

void Show()

{

cout << "Rectangle with verticles: ";

for(int i = 0; i < 4; i++)

cout << verticles[i].Get() << " ";

cout << endl;

}

void Input()

{

int x,y;

for(int i = 0; i < 4; i++)

{

cout << "Corner " << i+1 << ": ";

cin >> x >> y;

verticles[i].SetX(x);

verticles[i].SetY(y);

}

}

virtual float Perimeter()

{

return verticles[0].DistanceBetween(verticles[1]) + verticles[1].DistanceBetween(verticles[2])+

verticles[2].DistanceBetween(verticles[3]) + verticles[3].DistanceBetween(verticles[0]);

}

};

class Paralelogram : public Fourangle

{

public:

Paralelogram():Fourangle(){}

Paralelogram(Point &v1, Point &v2, Point &v3, Point &v4):Fourangle(v1, v2, v3, v4){}

float Perimeter() override { return Fourangle::Perimeter();}

};

class Romb: public Fourangle

{

public:

Romb():Fourangle(){}

Romb(Point &v1, Point &v2, Point &v3, Point &v4):Fourangle(v1, v2, v3, v4){}

float Perimeter() override { return Fourangle::Perimeter();}

};

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

Fourangle \*\*pubarr;

cout << "Write count: ";

int c;

cin >> c;

pubarr = new Fourangle\*[c];

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

{

cout << "Choose figure(Paralelogram(1), Romb(2)): ";

int type = 1;

cin >> type;

if(type == 2)

pubarr[i] = new Paralelogram();

else

pubarr[i] = new Romb();

pubarr[i]->Input();

}

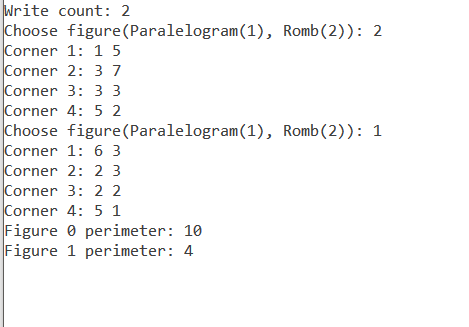
for(int i = 0; i < c; i++)

cout << "Figure " << i << " perimeter: " << pubarr[i]->Perimeter() << endl;

return a.exec();

}

Вивід програми:



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