Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Објектно ориентирано програмирање

Аудиториски вежби 12

Содржина

1. Виртуелен деструктор и задачи за вежбање

1.1. Виртуелен деструктор

1.1.1. Пример 1 – Зошто и кога е потребен виртуелен деструктор?

Решение без виртуелен деструктор `oop_av121a.cpp`

```
#include <iostream>
using namespace std;
class Osnovna {
public:
    Osnovna() { cout << "Konstruiram objekt od Osnovna\n";}
    // ова е деструктор:
    ~Osnovna() { cout << "Unishtuvam objekt od Osnovna\n";}
};
class Izvedena : public Osnovna
{
public:
    Izvedena() { cout << "Konstruiram objekt od Izvedena\n"; }
    ~Izvedena() { cout << "Unishtuvam objekt od Izvedena\n"; }
};
int main() {
    Osnovna *osnovnaPok = new Izvedena();
    delete osnovnaPok;
}
```

Излез од програмата

```
Konstruiram objekt od Osnovna
Konstruiram objekt od Izvedena
Unishtuvam objekt od Osnovna
```

Решение со виртуелен деструктор `oop_av121b.cpp`

```
#include <iostream>
using namespace std;
class Osnovna {
public:
    Osnovna() { cout << "Konstruiram objekt od Osnovna\n";}
    // ова е деструктор:
    virtual ~Osnovna() { cout << "Unishtuvam objekt od Osnovna\n";}
};
class Izvedena : public Osnovna
{
public:
    Izvedena() { cout << "Konstruiram objekt od Izvedena\n"; }
    ~Izvedena() { cout << "Unishtuvam objekt od Izvedena\n"; }
};
int main() {
    Osnovna *osnovnaPok = new Izvedena();
    delete osnovnaPok;
}
```

Излез од програмата

```
Konstruiram objekt od Osnovna
Konstruiram objekt od Izvedena
Unishtuvam objekt od Izvedena
Unishtuvam objekt od Osnovna
```

1.2. Втор колоквиум 2015/2016

1.2.1. Задача 1

Издавачката куќа FINKI-Education издава онлајн и печатени книги. За секоја книга се чуваат податоци за ISBN бројот (низа од најмногу 20 знаци), насловот (низа од најмногу 50 знаци), авторот (низа од најмногу 30 знаци) и основната цена изразена во $ (реален број). Класата за опишување на книгите е апстрактна (5 поени).

За секоја онлајн книга дополнително се чуваат податоци за url од каде може да се симне (динамички резервирана низа од знаци) и големината изразена во MB (цел број). За секоја печатена книга дополнително се чуваат податоци за масата изразена во килограми (реален број) и дали ја има на залиха (логичка променлива). (5 поени)

За секој објект од двете изведени класи треба да бидат на располагање следниве методи:

- Метод `bookPrice`, за пресметување на продажната цена на книгата на следниот начин: (10 поени)
  - За онлајн книга - цената се зголемува за 20% од основната цена ако книгата е поголема од 20MB
  - За печатена книга - цената се зголемува за 15% од основната цена ако масата на книгата е поголема од 0.7kg
- Преоптоварен оператор > за споредба на две книги од каков било вид според нивната цена. (5 поени)
- Преоптоварен оператор << за печатење на податоците за книгите во формат: (5 поени) `[ISBN]: [Наслов], [Автор] [Продажна цена]`

Да се имплементира функција mostExpensiveBook со потпис: `void mostExpensiveBook (Book** books, int n)` во која се печати вкупниот број на онлајн, односно, печатени книги во проследената низа посебно. (5 поени) Потоа се наоѓа и печати најскапата книга. (5 поени)

Да се обезбедат сите потребни функции за правилно функционирање на програмата. (5 поени)

Решение `oop_av122.cpp`

```
#include<iostream>
#include<string.h>
using namespace std;
class Book {
protected:
    char isbn[20];
    char title[50];
    char author[30];
    float price;
public:
    Book(const char* isbn = "", const char* title = "", const char* author = "", float
price = 0) {
        strncpy(this->isbn, isbn, 19);
        this->isbn[19] = 0;
        strncpy(this->title, title, 49);
        this->title[49] = 0;
        strncpy(this->author, author, 29);
        this->author[29] = 0;
        this->price = price;
    }
    void setISBN(char *isbn) {
        strncpy(this->isbn, isbn, 19);
        this->isbn[19] = 0;
    }
    virtual float bookPrice() = 0;
    char* getISBN() { return isbn; }
    friend ostream& operator<< (ostream& o, Book& b) {
        o << b.isbn << ": " << b.title << ", " << b.author << " " << b.bookPrice() <<
endl;
        return o;
    }
    virtual ~Book() {}
};

float Book::bookPrice() {
    return price;
}

bool operator>(Book& b1, Book& b2) {
    return (b1.bookPrice() > b2.bookPrice());
}

class OnlineBook : public Book {
private:
    char* url;
    int size;
public:
    OnlineBook(const char* isbn = "", const char* title = "", const char* author = "",
float price = 0, const char* url = "", int size = 0): Book(isbn, title, author, price) {
        this->url = new char[strlen(url) + 1];
        strcpy(this->url, url);
        this->size = size;
    }
    OnlineBook(OnlineBook& ob) {
        strcpy(isbn, ob.isbn);
        strcpy(title, ob.title);
        strcpy(author, ob.author);
```

```
        price = ob.price;
        url = new char[strlen(ob.url) + 1];
        strcpy(url, ob.url);
        size = ob.size;
    }
    OnlineBook& operator=(OnlineBook& ob) {
        if (this != &ob) {
            strcpy(isbn, ob.isbn);
            strcpy(title, ob.title);
            strcpy(author, ob.author);
            price = ob.price;
            delete[] url;
            url = new char[strlen(ob.url) + 1];
            strcpy(url, ob.url);
            size = ob.size;
        }
        return *this;
    }
    ~OnlineBook() {
        delete[] url;
    }
    float bookPrice() {
        if (size > 20)
            return Book::bookPrice() * 1.2;
        return Book::bookPrice();
    }
};

class PrintBook : public Book {
private:
    float weight;
    bool inStock;
public:
    PrintBook(const char* isbn = "", const char* title = "", const char* author = "",
float price = 0, float weight = 0, bool inStock = false): Book(isbn, title, author, price)
{
        this->weight = weight;
        this->inStock = inStock;
    }
    float bookPrice() {
        if (weight > 0.7)
            return Book::bookPrice() * 1.15;
        return Book::bookPrice();
    }
};

void mostExpensiveBook(Book** books, int n) {
    int obNo = 0;
    int pbNo = 0;
    for (int i = 0; i < n; i++)
    {
        OnlineBook* ob = dynamic_cast<OnlineBook*>(books[i]);
        if (ob != 0)
            obNo++;
        PrintBook* pb = dynamic_cast<PrintBook*>(books[i]);
        if (pb != 0)
            pbNo++;
    }
    cout << "FINKI-Education" << endl;
    cout << "Total number of online books: " << obNo << endl;
    cout << "Total number of print books: " << pbNo << endl;
    Book* max = books[0];
    for (int i = 1; i < n; i++)
        if (*books[i] > *max)
            max = books[i];
    cout << "The most expensive book is: " << endl;
    cout << *max;
}
```

1.3. Задача 2

Да се имплементира класа `Trud` во која се чуваат информации за: (5 поени)

- вид на труд (еден знак и тоа C за конференциски труд, J за труд во списание)
- година на издавање (цел број).

Да се имплементира класа `Student` во која се чува: (5 поени)

- името на студентот (низа од најмногу 30 карактери)
- индекс (цел број)
- година на упис (цел број)
- листа на оцени од положени предмети (низа од цели броеви)
- број на положени предмети (цел број).

За оваа класа да се имплементираат следните методи:

- функција rang() што пресметува просек од положените испити на студентот (5 поени)
- оператор << за печатење на студентот во формат: (5 поени) Индекс Име Година на упис ранг

Да се имплементира класа `PhDStudent` во која покрај основните информации за студентот дополнително се чува: (5 поени):

- листа од објавени трудови (динамички резервирана низа од објекти од класата `Trud`)
- бројот на трудови (цел број).

Во оваа класа да се препокрие соодветно функцијата `rang()` така што на просекот од положените испити ќе се додаде и збирот од поените од објавените трудови на PhD студентот. Во зависност од видот на трудот, секој универзитет има посебен начин на бодување на трудовите. Начинот на бодување е ист за сите PhD студенти. Иницијално да се смета дека конференциски труд се бодува со 1 поен, а труд во списание со 3 поени. Универзитетот има можност да ги менува вредностите на бодовите. (5 поени + 5 поени)

За оваа класа да се обезбеди:

- оператор += за додавање нов објект од класата Trud во листата (5 поени). Ако се направи обид да се внесе труд што е издаден порано од годината на упис на студентот да се фрли исклучок (објект од класата `Exception`). Справувањето со исклучокот треба да се реализира во главната функција main каде што е потребно, но и во конструктор ако е потребно. Ако бил генериран исклучок треба да се отпечати соодветна порака за грешка "Ne moze da se vnese dadeniot trud", а новиот труд нема да се внесе во листата на трудови од студентот. (10 поени)

Сите променливи на класите се чуваат како приватни.

Да се обезбедат сите потребни функции за правилно функционирање на програмата. (5 поени)

Решение `oop_av123.cpp`

```
#include<iostream>
#include<string.h>
using namespace std;
class Exception {
public:
    void print() {
        cout << "Ne moze da se vnese dadeniot trud" << endl;
    }
};
class Trud {
private:
    char tip;
    int god;
public:
    Trud(const char tip = 'C', int god = 0) {
        this->tip = toupper(tip);
        this->god = god;
    }
    int getGod() {
        return god;
    }
    char getTip() {
        return tip;
    }
    friend istream& operator>>(istream& in, Trud &t) {
        in >> t.tip >> t.god;
        return in;
    }
};
class Student {
private:
    char ime[30];
    int indeks;
    int god;
    int oceni[50];
    int n;
public:
    Student() {}
    Student(const char* ime, int indeks, int god, int *oceni, int n) {
        strcpy(this->ime, ime);
        this->indeks = indeks;
```

```
        this->god = god;
        this->n = n;
        for (int i = 0; i < n; i++)
            this->oceni[i] = oceni[i];
    }
    int getGod() {
        return god;
    }
    int getIndeks() {
        return indeks;
    }
    virtual float rang() {
        int suma = 0;
        for (int i = 0; i < n; i++)
            suma += oceni[i];
        return (float)suma / n;
    }
    friend ostream& operator<< (ostream& o, Student& st)
    {
        o << st.indeks << " " << st.ime << " " << st.god << " " << st.rang() << endl;
        return o;
    }
    virtual ~Student() {}
};
class PhDStudent : public Student {
private:
    Trud *t;
    int nt;
    static int conf;
    static int journal;
public:
    PhDStudent(const char* ime, int indeks, int god, int *oceni, int n, Trud* t, int nt) :
Student(ime, indeks, god, oceni, n) {
        //this->nt = nt;
        this->t = new Trud[100];
        int ok = 0;
        for (int i = 0; i < nt; i++) {
            try {
                if (this->getGod() > t[i].getGod()) throw Exception();
                this->t[ok] = t[i];
                ok++;
            }
            catch (Exception e) {
                e.print();
                //this->nt--;
            }
        }
        this->nt = ok
    }
    float rang() {
        int suma = 0;
        for (int i = 0; i < nt; i++) {
            if (t[i].getTip() == 'C')
                suma += conf;
            else
                suma += journal;
        }
        return Student::rang() + suma;
    }
    static void setConf(int c) {
        PhDStudent::conf = c;
    }
    static void setJournal(int j) {
        PhDStudent::journal = j;
    }
    void operator+=(Trud &tr) {
        if (this->getGod() > tr.getGod()) throw Exception();
        this->t[nt] = tr;
        this->nt++;
    }
    ~PhDStudent() {
        delete[] t;
    }
};
int PhDStudent::conf = 1;
int PhDStudent::journal = 3;
```

2. Изворен код од примери и задачи

https://github.com/finki-mk/OOP/

Source code ZIP