אביב 2017 מועד א - פתרון

פתרון שאלה 1:

פתרון סעיף 1:

א. נוכל להשתמש במצביע לפונקציה.  
ב. ניתן להגדיר לכל צורה ספציפית struct המכיל את המידע, ולשמור מצביע void\* למידע בתוך Shape.

פתרון סעיף 2:

**typedef** **struct** Shape\_ {

PDATA pShapeData;

COPY\_FUNC copyFunc;

DELETE\_FUNC deleteFunc;

DRAW\_FUNC drawFunc;

} Shape;

פתרון סעיף 3:

**typedef** **void**\* PDATA;

**typedef** **struct** Shape\_\* PShape;

/\*User functions\*/

**typedef** **void** (\*DRAW\_FUNC)(PDATA);

**typedef** PDATA (\*COPY\_FUNC)(PDATA);

**typedef** **void** (\*DELETE\_FUNC)(PDATA);

PShape **ShapeCreate**(DRAW\_FUNC drawFunc, COPY\_FUNC copyFunc, DELETE\_FUNC deleteFunc, PDATA pShapeData);

**void** **ShapeDraw**(PShape pShape);

**void** **ShapeDestroy**(PShape pShape);

פתרון סעיף 4:

**void** **ShapeDraw**(PShape pShape){

**if**(pShape == NULL){

**return**;

}

pShape->drawFunc(pShape->pShapeData);

}

פתרון סעיף 5:

לא ניתן בתצורה הנוכחית של Shape כאלמנט ברשימה המקושרת.

על מנת שנוכל לבצע זאת, המודול Shape צריך לספק פונקציית ממשק לשכפול Shape.

|  |
| --- |
| PShape **ShapeClone**(PShape pShape){  **if**(pShape == NULL){  **return** NULL;  }  PShape pNewShape = (PShape)**malloc**(**sizeof**(\*pNewShape));  **if**(pNewShape == NULL){  **return** NULL;  }  pNewShape->drawFunc = pShape->drawFunc;  pNewShape->deleteFunc = pShape->deleteFunc;  pNewShape->copyFunc = pShape->copyFunc;  pNewShape->pShapeData = pNewShape->copyFunc(pNewShape->pShapeData);  **return** pNewShape;  } |

**שאלה 2 פתרון**

#include <vector>

#include <algorithm>

#include <iostream>

using namespace std;

class BankAccount {

public:

BankAccount(int id = -1, double balance = 0) :

id\_(id), balance\_(balance) {

}

virtual BankAccount& operator+=(double sum) = 0;

friend ostream& operator<<(ostream& ro, const BankAccount& acc);

friend class compare\_accounts;

protected:

int id\_;

double balance\_;

virtual void print(ostream& ro) const = 0;

};

ostream& operator<<(ostream& ro, const BankAccount& acc) {

acc.print(ro);

return ro;

}

class LocalAccount: public BankAccount {

public:

LocalAccount(int id = -1, double balance = 0) :

BankAccount(id, balance) {

}

BankAccount& operator+=(double sum) {

this->balance\_ += sum;

return \*this;

}

virtual void print(ostream& ro) const {

ro << "id: " << id\_ << " , balance: " << balance\_;

}

};

class ForeignAccount: public BankAccount {

public:

ForeignAccount(double exchangeRate, int id = -1, double balance = 0) :

BankAccount(id, balance), exchangeRate\_(exchangeRate) {

}

BankAccount& operator+=(double sum) {

this->balance\_ += sum \* exchangeRate\_;

return \*this;

}

virtual void print(ostream& ro) const {

ro << "id: " << id\_ << " , balance: " << balance\_ << " , ex rate: " << exchangeRate\_;

}

private:

double exchangeRate\_;

};

class compare\_accounts {

public:

bool operator()(BankAccount\* a, BankAccount\* b) {

return (a->id\_ < b->id\_);

}

};

class Bank {

public:

void addAccount(BankAccount\* acc) {

accounts.push\_back(acc);

}

void printAccounts() {

sort(accounts.begin(), accounts.end(), ca);

for (vector<BankAccount\*>::iterator it = accounts.begin();

it != accounts.end(); ++it) {

cout << \*(\*it) << endl;

}

}

private:

vector<BankAccount\*> accounts;

compare\_accounts ca;

};

שאלה 3 **פתרון:**

**סעיף א':**

|  |
| --- |
| **class** A{  **public**:  **virtual** **~A**(){cout << "~A" <<**endl**;}  **int** **foo**(**const** **int**& num){**return** num;}  };  **class** B: **public** A{  **public**:  A\* a;  **B**(){a = **new** A();}  **~B**(){**delete** a;}  }; |

1. אתחול של רפרנס יכול להתבצע רק על ידי lvalue(ביטוי שמופיע בצד שמאל של ההשמה) כלומר לא ניתן לאתחל רפרנס רגיל למשתנה קבוע ולכן יש להוסיף את המיל השמורה const.

2. יש להוסיף הורס וירטואלי לA אחרת לא נקרא ההורס של B כאשר מבצעים **delete** array[1];.

**סעיף ב':**

|  |
| --- |
| **class** Singleton  {  **private**:  **static** **bool** *instanceFlag*;  **static** Singleton \**single*;  **Singleton**() {}  **static** **int** *counter*;  **public**:  **static** Singleton\* **getInstance**();  **static** **int** **getNumInstances**();  **static** **void** **removeInstance**();  **~Singleton**()  {  *instanceFlag* = **false**;  *counter*--;  }  };  **bool** *Singleton::instanceFlag* = **false**;  **int** *Singleton::counter* = 0;  Singleton\* *Singleton::single* = NULL;  Singleton\* **Singleton::getInstance**()  {  **if**(! *instanceFlag*)  {  *single* = **new** Singleton();  *instanceFlag* = **true**;  *counter*++;  **return** *single*;  }  **else**  {  **return** *single*;  }  }  **void** **Singleton::removeInstance**()  {  **if**(*instanceFlag*)  {  **delete** *single*;  }  }  **int** **Singleton::getNumInstances**()  {  **return** *counter*; |

**שאלה 4 פתרון**

א. פתרון שאלת BASH

א.

grep -v details names|grep $1| cut -f3 -d" " |cut -f2 -d"/"

grep -v details names|cut -f3 -d" " |cut -f2 -d"/" |./ find

find:

#! /bin/bash

sum=0

n=0

while read line; do

(( n +=1 ))

sline=($line)

(( sum += ${sline[0]} +0 ))

done

(( average=$sum/$n ))

echo $average

ב. פתרון שאלות הבנה

1. בשפת C : פונקציה סטטית

בשפת C++: unnamed namespace

1. בעזרת אובייקט פונקציה המשווה מחרוזות, אותו ניתן כפרמטר נוסף לקונסטרקטור של הסט.
2. לא ניתן להחזיר מהפונקציה משתנה פנימי be reference מאחר והוא לא קיים יותר כשיוצאים מפונקציה.