**אביב 2017 מועד ב - פתרון**

פתרון שאלה 1:

פתרון סעיף 1:

**typedef** **struct** Animal\_ {

MAKE\_SOUND\_FUNC makeSoundFunc;

} Animal, \*PAnimal;

פתרון סעיף 2:

כן

**void** **AnimalDestroy**(PAnimal pAnimal){

**if**( pAnimal == NULL ){

**return**;

}

**free**(pAnimal->name);

**free**(pAnimal);

}

פתרון סעיף 3:

**typedef** **struct** Cat\_ {

PAnimal pAnimal;

**int** meowPerHour;

} Cat, \*PCat;

פתרון סעיף 4:

PCat **catConstructor**(MAKE\_SOUND\_FUNC makeSoundFunc, **char**\* name, **int** meowPerHour){

**if**(makeSoundFunc == NULL){

**return** NULL;

}

PCat pCat = (PCat)**malloc**(**sizeof**(\*pCat));

**if**(pCat == NULL){

**return** NULL;

}

pCat->pAnimal = AnimalCreate(makeSoundFunc, name);

**if**(pCat->pAnimal == NULL){

**free**(pCat);

**return** NULL;

}

pCat->meowPerHour = meowPerHour;

**return** pCat;

}

פתרון סעיף 5:

/\*User functions\*/

**typedef** BOOL (\*COMP\_FUNC)(PElem, PElem);

**typedef** PElem (\*CLONE\_FUNC)(PElem);

**typedef** **void** (\*DESTROY\_FUNC)(PElem);

פתרון סעיף 6:

|  |
| --- |
| BOOL **catCompare**(PElem pElem1, PElem pElem2){  **if**(pElem1 == NULL || pElem2 == NULL){  **return** *FALSE*;  }  PCat pCat1 = (PCat)pElem1;  PCat pCat2 = (PCat)pElem2;  **if**(pCat1->meowPerHour == pCat2->meowPerHour){  **return** *TRUE*;  }  **return** *FALSE*;  } |

**שאלה 2 - פתרון**

#include <vector>

using namespace std;

class SafeVecException {

};

template<class T>

class SafeVec {

public:

SafeVec() : max\_(0), pvector\_(new vector<T>(max\_)) {};

SafeVec(unsigned max) : max\_(max), pvector\_(new vector<T>(max\_)) {};

T& operator[](unsigned int i);

const T& operator[](unsigned int i) const;

SafeVec& operator=(const SafeVec& rhs);

SafeVec(const SafeVec& rhs) : max\_(rhs.max\_), pvector\_(new vector<T>(\*rhs.pvector\_)) {};

~SafeVec();

class SafeIterator {

public:

SafeIterator(typename vector<T>::iterator val) : iter\_(val) {};

SafeIterator& operator++() {

iter\_++;

return \*this;

}

T& operator\*() {

return \*iter\_;

}

bool operator!=(const SafeIterator& rhs) {

return iter\_ != rhs.iter\_;

}

private:

typename vector<T>::iterator iter\_;

};

SafeVec::SafeIterator begin() {

return SafeIterator((\*pvector\_).begin());

}

SafeVec::SafeIterator end() {

return SafeIterator((\*pvector\_).end());

}

private:

unsigned int max\_;

vector<T>\* pvector\_;

};

template<class T>

T& SafeVec<T>::operator[](unsigned int i) {

if (i >= max\_) {

SafeVecException e;

throw e;

}

return (\*pvector\_)[i];

}

template<class T>

const T& SafeVec<T>::operator[](unsigned int i) const {

if (i >= max\_) {

SafeVecException e;

throw e;

}

return (\*pvector\_)[i];

}

template<class T>

SafeVec<T>& SafeVec<T>::operator=(const SafeVec<T>& rhs) {

if (this != &rhs) {

max\_ = rhs.max\_;

delete pvector\_;

pvector\_ = new vector<T>(\*rhs.pvector\_);

}

return \*this;

}

template<class T>

SafeVec<T>::~SafeVec() {

delete pvector\_;

}

SafeIterator(typename vector<T>::iterator val) : iter\_(val) {};

SafeIterator& operator++() {

iter\_++;

return \*this;

}

T& operator\*() {

return \*iter\_;

}

bool operator!=(const SafeIterator& rhs) {

return iter\_ != rhs.iter\_;

}

private:

typename vector<T>::iterator iter\_;

};

SafeVec::SafeIterator begin() {

return SafeIterator((\*pvector\_).begin());

}

SafeVec::SafeIterator end() {

return SafeIterator((\*pvector\_).end());

}

private:

unsigned int max\_;

vector<T>\* pvector\_;

};

template<class T>

T& SafeVec<T>::operator[](unsigned int i) {

if (i >= max\_) {

SafeVecException e;

throw e;

}

return (\*pvector\_)[i];

}

template<class T>

const T& SafeVec<T>::operator[](unsigned int i) const {

if (i >= max\_) {

SafeVecException e;

throw e;

}

return (\*pvector\_)[i];

}

template<class T>

SafeVec<T>& SafeVec<T>::operator=(const SafeVec<T>& rhs) {

if (this != &rhs) {

max\_ = rhs.max\_;

delete pvector\_;

pvector\_ = new vector<T>(\*rhs.pvector\_);

}

return \*this;

}

template<class T>

SafeVec<T>::~SafeVec() {

delete pvector\_;

}

**שאלה 3 פתרון**

#include <stdlib.h>

#include <string.h>

class FamilyMember {

public:

FamilyMember() :

name\_(NULL) {

members\_++;

}

FamilyMember(const char\* name) {

members\_++;

if (name == NULL) {

return;

}

name\_ = new char[strlen(name) + 1];

strcpy(name\_, name);

}

FamilyMember(const FamilyMember& rhs) {

members\_++;

if (rhs.name\_ == NULL) {

return;

}

name\_ = new char[strlen(rhs.name\_) + 1];

strcpy(name\_, rhs.name\_);

}

FamilyMember& operator=(const FamilyMember& rhs) {

if (this != &rhs) {

if (name\_ != NULL) {

delete name\_;

}

name\_ = new char[strlen(rhs.name\_) + 1];

strcpy(name\_, rhs.name\_);

}

return \*this;

}

static const int getNumOfMembers() {

return members\_;

}

static void setFamilyName(const char\* name) {

if (name == NULL) {

return;

}

familyName\_ = new char[strlen(name) + 1];

strcpy(familyName\_, name);

}

static const char\* getFamilyName() {

return familyName\_;

}

const char\* getName() {

return name\_;

}

~FamilyMember() {

members\_--;

delete[] name\_;

if (members\_ == 0) {

delete[] familyName\_;

}

}

private:

static int members\_;

static char\* familyName\_;

char\* name\_;

};

int FamilyMember::members\_ = 0;

char\* FamilyMember::familyName\_ = NULL;

**פתרון שאלה 4**

פתרון שאלות BASH

1.

#!/bin/bash

grep using \*.[cC] |grep namespace| wc -l

2.

check

#! /bin/bash

cat example | check1

check1

#!/bin/bash

count=`wc -l example`

scount=($count)

while read line; do

sline=($line)

words=${#sline[@]}

if (( $words != ${scount[0]} )); then

echo "not NXN"

exit 0

fi

done

echo ${scount[0]}

פתרון שאלות ההבנה

1. פונקציה רגילה מספקת פתרון ספציפי המתאים לטיפוס מסויים. פונקציה טמפלייטית מספקת פתרון כללי מידי שיכול לא להתאים למקרה ספציפי. למשל הפונקציה הנתונה קוראת לפונקציית השוואה המשווה בין הערכים של הטיפוס הנתון. במקרה של השוואת char \* היא תשווה בין הפוינטרים ואילו הפונקציה הספציפית תשתמש ב strcmp.
2. תופסת את כל החריגות שנורשו מ exception.

מנגנון פולימורפיזם שיקרא לפונקציה הדורסת את what() ויחזיר מחרוזת המתארת את החריגה הספציפית.

1. מנגנון inline חוסך את התקורה בזמן ריצה של קריאה לפונקציה, בכך שהוא עובד כמו

מקרו ומבצע החלפת קוד בזמן קומפילציה.