המסלול האקדמי
המכללה למינהל

להצלחה יש דרך

מבחן לדוגמא - תכנות מונחה עצמים בשפת C++. תשע"ה.

תקציר

מבחן זה מהווה דוגמא למבחן בפורמט החדש. מבחן זה בוחן את יכולות התכנות שלכם בשפה C++.
בפרט, מבחן זה בוחן את היכולת שלכם לנצל את הידע שרכשתם בקורס כדי לפתור בעיות תכנות בצורה מונחית עצמים ובצורה גנרית.

אליהו חלסצ'י
eliahukh@colman.ac.il

הקדמה

מבחן זה מהווה דוגמא למבחן בפורמט החדש. מבחן זה בוחן את יכולות התכנות שלכם בשפה ++C. בפרט, מבחן זה בוחן את היכולת שלכם לנצל את הידע שרכשתם בקורס כדי לפתור בעיות תכנות בצורה מונחית עצמים ובצורה גנרית.

מטרת המבחן-לדוגמא היא ליצור מעט וודאות לקראת המבחן האמתי. מספר השאלות והנושאים הנשאלים יהיו דומים, אך בטח שלא זהים(!). מספר הסעיפים והתפלגות הנקודות בהחלט עשויים להשתנות ממבחן למבחן. הנושאים הנשאלים יהיו דומים אך בכל מבחן ייבחנו אספקטים שונים של נושאים אלה.

כדי להצליח במבחן האמתי יש ללמוד את החומר; ולא את המבחן לדוגמא בלבד. לפתור מבחנים משנים קודמות עבור התרגול של החומר בהחלט יעזור לכם להתכונן למבחן. שימו לב שבשנים קודמות לא נבחנו הסטודנטים על קבצים ולא על reference counting והמבחנים הבאים בהחלט עשויים לבחון את הידע שלכם בנושאים אלה.

לאורך כל המבחן בדיקת הקוד שלכם תכלול:

- סינטקס נכון
- const, והעברת פרמטרים נכונה
- ייצוג מינימאלי ללא קוד כפול

פרטים טכניים:

- המבחן עם חומר סגור. המבחן עשוי להכיל רמזים או תזכורות במידת הצורך.
- יש לענות אך ורק בטופס המבחן. מחברות הטיוטה לא ייבדקו.
- אין שאלות בחירה.
- משך הבחינה – 3 שעות. אלמנט הזמן הוא חלק מהמבחן. קוד גנרי החוסך קוד כפול יחסוך לכם זמן בבחינה.
- 6 שאלות בנושאים:
  - ירושה והכלה
  - בנאים \ הורסים
  - פולימורפיזם
  - קבצים
  - General container / iterator / templates
  - Generic algorithm

רצוי לעבור על המבחן מתחילתו ועד סופו לפני שמתחילים לפתור, ולתכנן את הזמן לכל שאלה.

בהצלחה.

שאלה 1 – ירושה והכלה

הגדרת מחלקה מורכבת משם המחלקה, מה היא יורשת, וה data members שלה. לדוגמא:

```
class B: public A {
        int x,y; //...
```

ברצוננו ליצור משחק מחשב. הגדירו את המחלקות הבאות, מותר ואף רצוי להוסיף מחלקות עזר בעת הצורך וע"פ הגיון. עבור השדה "שם" יש להשתמש ב *char.

א. המחלקה Warrior עבור דמות לוחם. ללוחם יש:
   a. שם, מיקום (x,y), מד חיים (0-5), מגן (Shield) וחרב (Sword)
ב. המחלקה Medic עבור דמות מרפא. למרפא יש:
   a. שם, מיקום (x,y), מד חיים (0-5), כמות חומר מרפא (0-100)
ג. המחלקה Warlock עבור דמות מכשף. למכשף יש:
   a. שם, מיקום (x,y), מד חיים (0-5), כמות חומר כישוף (0-100)
ד. המחלקה MedicWarrior עבור דמות מרפא שהוא גם לוחם.

תשובה:

בשאלה זו עלינו לחסוך קוד ע"י שימוש בירושה. כל הדמויות במשחק הן קודם כל דמויות, לכולן יש את המידע המשותף: שם, מיקום (x,y), ומד חיים (0-5). לכן נתחיל מהגדרת מחלקה עבר דמות משחק. נשים לב שהמיקום מורכב מכמה משתנים. בתכנות מונחה עצמים אנו לא רוצים לעבוד עם שברים, אלא עם אובייקטים שמאחדים מידע. ולכן גם נגדיר מחלקה עבור מיקום. ובכלל בשאלות מסוג זה כדאי להבחין בין ירושה להכלה.

```
class Position{
        int x,y; //...
};
class GameCharacter{
        char* name;
        Position p;
        char lifeLevel; //...
};
```

כעת נוכל לענות על הסעיפים השונים. **בסעיף א'** עלינו ליצור מחלקות עבור המגן והחרב. מחלקת הלוחם תירש את דמות המשחק ותכיל את המגן והחרב.

```
class Sword{ //...};
class Shield{ //...};
class Warrior : virtual public GameCharacter{
        Sword mySword;
        Shield myShield; //...
};
```

סעיף ב':

```
class Medic : virtual public GameCharacter{

        int healingLevel; //…

};
```

סעיף ג':

```
class Warlock : virtual public GameCharacter{

        int magicLevel; //…

};
```

סעיף ד': כאן עלינו פשוט להשתמש בירושה מרובה. אך נשים לב שלא נרצה שהחלק של GameCharacter יופיע פעמיים עבור אובייקט מסוג זה; הרי לדמות כזו יש רק שם אחד, מיקום אחד וכו'. לשם כך נרצה בסעיפים הקודמים שהמחלקות היורשות את GameCharacter ירשו אותה בצורה וירטואלית. זה נכון לנהוג כך עבור כל המחלקות ולא רק עבור Medic ו Warrior שכן ייתכנו בעתיד שילובים נוספים וכלל זה יהיה נכון גם עבורם.

```
class MedicWarrior: public Medic, public Warrior {//…};
```

שאלה 2 – constructors / destructors

כתבו בנאי מונחה עצמים עבור המחלקה MedicWarrior. במידת הצורך כתבו בנאים נוספים עבור מחלקות נוספות.

תשובה:

בנאי מונחה עצמים הוא בנאי שמקבל כפרמטרים את האובייקטים הנדרשים כדי לאתחל את ה data members של המחלקה. בירושה עלינו לדאוג לאתחל את החלק שירשנו. בירושה מרובה עלינו לדאוג לאתחל גם את האב הקדמון. נשים לב אם יש צורך בשורת אתחול, או copy CTOR.

```
MedicWarrior:: MedicWarrior(const Medic& medic, const Warrior& warrior) :
GameCharacter(medic) ,  Medic(medic), Warrior(warrior) {}

GameCharater:: GameCharater(const GameCharacter& character){

        name=new char[strlen(character.name)+1];

        strcpy(name, character.name);

        p= character.p;

        lifeLevel= character.lifeLevel;

}

Medic::Medic(const Medic& medic) : GameCharacter(medic),
healingLevel(medic.healingLevel) {}

Warrior:: Warrior(const Warrior& warrior) : GameCharacter(warrior),
mySowrd(warrior.mySword), myShield(warrior.myShield) {}
```

שאלה 3 – פולימורפיזם

לכל דמות משחק יש כוח המחושב באופן הבא:

- כוח הלוחם – מד החיים X 3
- כוח המרפא – מד החיים X 2
- כוח המכשף – מד החיים X 5
- **(כוח הלוחם המרפא - מד החיים X 4)

במחלקה Shield נוספה המתודה הבאה המחזירה את חוזק המגן:

```
int getStrength() const;
```

ממשו פונקציה בשם sumTotalStrength המקבלת מערך שיכול להכיל כל דמות משחק, ומחזירה סכום הכולל את כוחה של כל דמות + חוזקם של כל המגנים. ממשו מתודות עזר (מונחות עצמים) במידת הצורך.

תשובה:

ראשית כל, נעזר במתודה וירטואלית במחלקה GameCharacter:

```
public:

virtual int getStrength()=0;
```

במחלקות האחרות נממש את המתודה הזו בהתאם להגדרה, לדוג':

```
int Warrior::getStrength() { return lifeLevel*3 }
```

**נשים לב שגם ללא הגדרה, היה חובה עלינו לממש את המתודה getStrength במחלקה MedicWarrior. אחרת, היינו מקבלים שגיאת דו-משמעות בגלל הירושה המרובה שלה.

נמשיך. רק ללוחמים יש מגנים. במחלקה Warrior נצטרך להוסיף מתודה המחזירה את המגן שלו. מתוך המגן שחזר נוכל לחלץ את החוזק. נשים לב שלא נממש מתודה בתוך המחלקה Warrior שמחזירה את חוזק המגן, שכן חוזק המגן היא הפונקציונאליות של המגן, לא של הלוחם. כמו כן, לא נממש במתודה getStrength של הלוחם את הסכום של חוזק המגן + חוזק הלוחם, שכן אלו שתי תכונות נפרדות.

```
const Shield& Warrior::getShield() const { return myShield; }
```

כעת נוכל לממש את הפונקציה שלנו. מערך שיכול להכיל כל דמות משחק חייב להיות כזה שכל תא הוא מסוג *GameCharacter שכן מצביע כזה יכול להצביע לכל סוג של GameCharacter. המערך שלנו יהיה מסוג **GameCharacter.

נקודה שניה שיש לשים לב אליה היא שבעת המעבר על כל תא, נצטרך לזהות שמדובר באובייקט מסוג לוחם. לא נוכל להשתמש ב typeid(*array[i])==typeid(Warrior) בלבד, שכן לא נגלה כך אובייקטים מסוג MedicWarrior. במקום לממש השוואה נוספת, הרבה יותר אלגנטי להשתמש ב dynamic casting.

```
int sumTotalStrength(GameCharacter** array){
	int count=0;
	for(int i=0;i< sizeof((array))/sizeof((array[0]);i++){
		count+=array[i]->getStrength();
		Warrior* w=dynamic_cast<Warrior*>(array[i]);
		If(w)
			count+=w->getShield()->getStrength();
	}
	return count;
}
```

שאלה 4 – קבצים

א. ממשו במחלקה Medic את המתודה save המקבלת אובייקט מסוג ofstream ובאמצעותו היא שומרת את נתוני המרפא בקובץ.

ב. כמו כן, ממשו את המתודה load המקבלת אובייקט מסוג ifstream שבאמצעותו היא טוענת את נתוני המרפא מתוך קובץ.

ממשו מתודות עזר נוספות במחלקות אחרות במידת הצורך.

תשובה:

את רוב העבודה נצטרך לעשות במחלקה GameCharacter כי שם נמצאים רוב הנתונים. נקפיד שהמתודות save & load יהיו virtual. במחלקה Medic נשתמש במתודות שירשנו. נזכור שעבור הקצאה דינאמית נצטרך לשמור תחילה את גודל ההקצאה ואז את התוכן שלה.

```
void Position::save(ofstream& out) const {
    out.write((char*) &x, sizeof(x));
    out.write((char*) &y, sizeof(y));
}
// in GameCharacter
virtual void save(ofstream& out) const{
    int n = strlen(name);
    out.write((char*) &n, sizeof(n));
    out.write(name, n);
    p.save(out);
    out.write((char*) &lifeLevel, sizeof(lifeLevel));
}
```

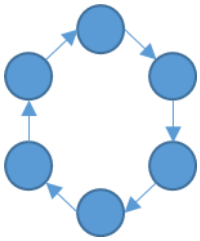
```
void Medic::save(ofstream& out) const {
	GameCharacter::save(out);
	out.write((char*) & healingLevel, sizeof(healingLevel));
}
```

סעיף ב':

```
void Position::load(ifstream& in){
	in.read((char*)&x, sizeof(x));
	in.read((char*)&y, sizeof(x));
}
// in GameCharacter
virtual void load(ifstream& in){
	int n;
	in.read((char*)&n, sizeof(n));
	name = new char[n + 1];
	in.read(name, n);
	name[n] = '\0';
	p.load(in);
	in.read((char*)&lifeLevel, sizeof(lifeLevel));
}
void Medic::load(ifstream& in){
	GameCharacter::load(in);
	in.read((char*)&healingLevel, sizeof(healingLevel));
}
```

שאלה 5 – templates \ iterator \ generic container

רשימה מעגלית היא מבנה נתונים המורכב מרשימה מקושרת חד כיוונית כאשר אין לה התחלה או סוף, האיבר הבא של האיבר "האחרון" הוא לא אחר מאשר האיבר "הראשון".



ברצוננו לממש את המחלקה RoundedLinkedList עבור מבנה נתונים זה, כך שתוכל להכיל כל סוג של נתונים.

א. ממשו מחלקת עזר בשם Item המייצגת איבר במבנה נתונים זה
ב. ממשו iterator עבור מבנה נתונים זה

תשובה:

א. כדי שמבנה הנתונים יוכל להכיל כל סוג של נתון המחלקה של Item תצטרך להיות תבנית למחלקה. כל Item מייצג איבר עם קישור יחיד לאיבר הבא. נזכור שמחלקה זו נועדה לכך שלא נצטרך לשנות את המחלקה המייצגת את הנתון הנשמר.

```
template<class T> class Item{
	Item<T>* next;
	T* data;
	public:
	Item(const T& aData):data(new T(aData)),next(NULL){}
	void setNext(Item<T>* aNext) { next=aNext}
	Item<T>* getNext() const { return next; }
	T* getData(){ return data;}
	~Item(){ delete data;}
};
```

ב. ה Iterator שלנו ימומש בתוך המחלקה RoundedLinkedList שכמובן תהיה template. הוא יצטרך לממש את האופרטורים ++, !=, ==, *

```
template<class T> class RoundedLinkedList{ //…
	class iterator{
		Item<T>* p;
		public:
		iterator(Item<T>* aP):p(aP){}
		iterator operator++() throw (char*){
			if(p==NULL) throw “NULL pointer exception!”;
			p=p->getNext();
			return this;
		}
		int operator==(iterator it){ return (p==it.p); }
		int operator!=(iterator it){ return (p!=it.p); }
		T& operator*() throw (char*) {
			if(p!=NULL && p->getData()!=NULL)
				return *(p->getData());
			else throw “NULL pointer exception!”;
		}
	};
	//…
};
```

שאלה 6 – generic algorithm

א. צרו מופע של RoundedLinkedList השומרת int-ים
ב. צרו מופע של RoundedLinkedList השומרת double-ים
ג. מופע של RoundedLinkedList השומרת דמויות משחק
ד. ממשו פונקציה שמדפיסה את כל האיברים במבנה נתונים כלשהו
ה. ממשו פונקציה ששומרת בקובץ את כל האיברים במבנה נתונים כלשהו
ו. שימרו את כל דמויות המשחק בקובץ "characters.bin".

תשובה:

```
RoundedLinkedList<int> intsList;

RoundedLinkedList<double> doublesList;

RoundedLinkedList<GameCharacter*> charactersList;
```

עבור סעיפים ד' וה' נוכל להשתמש באותה הפונקציה הכללית בעזרתו של Object Function כלשהו:

```
template<class iterator, class func>

void apply(iterator begin, iterator end,const func& f){

	for(;begin!=end;begin++){ f(*begin); }

}
```

עבור סעיף ד' נממש:

```
struct Print{

	template<class Printable>

	void operator(){const Pritnable& printable){ cout<<p<<endl; }

};
```

עבור סעיף ה' נממש:

```
struct Save{

	ofstream& out;

	Save(ofstream& aOut):out(aOut){}

	template<class Savable>

	void operator(){const Savable& savable){ savable.save(out); }

};
```

סעיף ו':

```
apply(charactersList.begin(), charactersList.end(), Save(ofstream("characters.bin")));
```