1/8 אמיר

קורס 0368-3058-01 - נושאים מתקדמים בתכנות: 11/07/2010 09:00 א' כ"ט תמוז תש"ע

ציון: 92

מס' מחברת 63

מרצה –יוסף הלחמי

משך המבחן – 3 שעות
אין להשתמש בחומר עזר

חלק א' - 8 שאלות אמריקאיות 5 נק' לשאלה, ס"הכ 40 נקודות.

40

	א	ב	ג	ד
שאלה 1			X	
שאלה2	X			
שאלה 3	X			
שאלה 4			X	
שאלה 5	X			
שאלה 6				X
שאלה 7		X		
שאלה 8				X

שאלה 1 מה קורה בתוכנית הבאה?

```
                    void   func(int)  { cout << "func(int)" << endl;  }
template <class T>  void   func(T t)  { cout << "func(T)" << endl;    }

int main() {
    int i = 4;
    func(i);
}
```

א תוכנית לא מתקמפלת -ambiguity.
ב אם מ T יש המרה ל INT אז פלט - "func(int)"
ג פלט - "func(int)"
ד פלט - "func(T)" ; אחרת : פלט - "func(T)"

שאלה 2 מה ההבדל בין מצביע (POINTER) ו-REFERENCE ב-C++?

א Reference הוא האובייקט כולו בעוד מצביע רק את הכתובת שלו.

ב משמעות זהה, וההבדל הוא רק התחביר(syntax) והשימוש.

ג התחביר המשמש לגישה האובייקט.

ד מצביע הוא הכתובות לאובייקט ו-Reference משתמש בטבלה הוירטואלית.

שאלה 3. Exception Handler:

א משמיד כל locally-declared אובייקטים שנוצרו בכל פונקציות שדרכן Exception Handler עובר.

ב משמיד כל האובייקטים דינמיים שנוצרו על ה-HEAP כל פונקציות שדרכך Exception Handler עובר.

ג תשובות א' וגם ב' נכונות

ד Exception Handler לא הורס האובייקטים שנוצרו כל בפונקציה שדרכן הוא עובר.

שאלה 4 מה המטרה לעשות destructor "VIRTUAL"?

א אין שום ערך.

ב משתמשים בו לאיתור דליפת זיכרון.

ג זה גורם לdestructor של המחלקה הנגזרת להקרא לפני אופרטור delete.

ד destructor של מחלקה נגזרת נקרא תמיד, זה פשוט משנה את הסדר של קריאות של DTORs.

שאלה 5 איזה תיאור של **pure virtual function** הוא נכון?

א היא מכריזה לרשת ממחלקת הבסיס המכילה פונקציה הזאת.

ב. אין להן ישום (implementation).

ג תשובות א' וב' לא נכונות

ד תשובות א' וב' נכונות

שאלה6 תבנית עיצוב "**Singleton**":

א יוצר רק אובייקט אחד מסוג מסוים

ב מחלקה היורשת היא לא Singleton

ג עדיף להשתמש תמיד במקום משתנים גלובליים

ד כל התשובות נכונות.

שאלה7 תבנית עיצוב "**Factory Method**":

א יוצר אובייקטים חדשים על ידי עתקה ושיבוט של אובייקט אב איפוס (cloning)

ב מגדיר ממשק ליצירת אובייקטים, אבל מחלקות יורשות מחליטות איזה אובייקט לבנות בפועל.

ג הגדרת התלויות בין אובייקטים, כך שכאשר אובייקט אחד עובר שינויים , אז התלויים בו מקבלים הודעה על זה באופן אוטומטי.

ד כל התשובות נכונות.

שאלה8 מהו המספר המינימלי של data-members שיכול להיות במחלקה ?

א מינימום אחד - מוגדר על ידי המתכנת

ב מינימום אחד - מצביע על טבלה וירטואלית (מוגדר על ידי קומפיילר)

ג מינימום שני, הראשון המוגדר על ידי המתכנת, והשני הוא מצביע על טבלה וירטואלית

ד אפס

חלק ב' - 8 שאלות אמריקאיות 5 נק' לשאלה, ס"הכ 40 נקודות.

```
1.  #include <iostream>
2.  using namespace std;

3.  class Person { //…};

4.  class Base {
5.  public:
6.      virtual     ~Base() {  delete m_ptr; }
7.                  Base() {  m_ptr = new int(7); }
8.                  Base(int k) : m_int(k), m_ptr(m_int) { foo();}
9.      virtual void foo() = 0;
10.     virtual void bar() { cout << m_int     << endl;}
11. protected:
12.                 Base() {  m_ptr = new int(7); }
13. private:
14.     int  m_int;
15.     int* m_ptr;
16. };

17. void Base::foo() { cout << *(m_ptr)     << endl;}
18.
19. class Drv: public Base {
20. public:
21.     Drv (int i, char c) : m_c1(c), m_c2(c), Base(i) {foo(); }
22.     void      bar() {
23.         cout << "in Drv::bar() " << endl;
24.         Base::bar();
25. };
26.     void      foo() { cout << m_c1 << endl;}

27. private:
28.     char      m_c1, m_c2;
29.
30.     friend class Person;
31. };

32. int main() {
33.     Drv   d1;
34.     Drv   d2(55, 'a');
35.     Base* b1 = &d1;
36.     Base* b2 = &d2;
37.     *b2 = *b1;
38.     b2->foo();
39.     b2->bar();
40.     return 0;
41. }
```

שם מתכנת: 63

ציון 4/8

שאלה 1:

בשורה 21 לאיזה פונקציה FOO פונה קומפיילר?

Drv (int i, char c) : m_c2(c), Base(i), m_c1(c) {foo(); }

א) לפונקציה FOO של מחלקה Drv מפני שFOO היא וירטואלית
ב) לפונקציה FOO של מחלקה Base מפני שCTOR של Drv קודם פונה לCTOR של Base
ג) קודם לFOO של Base ואחר כך לFOO של Drv
ד) קודם לFOO של Drv ואחר כך לFOO של Base
ה) אף תשובה אינה נכונה

שאלה 2:

שורה 17: מה יקרה אם נשים אותה בהערה ?

// void Base::foo() { cout << *(m_ptr) << endl;};

א) שגיאת קומפילאציה כי CTOR של BASE פונה אליה וחייבים לממש אותה
ב) שגיאת קומפילאציה כי PURE VIRTUAL FUNCTION תמיד חייבת במימוש וגם מחייבת מימוש במחלקה יורשת
ג) אין שגיאת קומפילאציה כי PURE VIRTUAL FUNCTION לא מחייבת מימוש
ד) אין שגיאת קומפילאציה כי יש מימוש במחלקת Drv
ה) אף תשובה אינה נכונה

שאלה 3:

שורה 12 – protected CTOR

Base() { m_ptr = new int(7); }

א) אי אפשר לרשת ממחלקה Base כי בשביל זה CTOR של Base חייב להיות public
ב) תמיד אפשר לרשת ממחלקה Base אפילו אם CTOR של Base protected או private
ג) תמיד אפשר לרשת ממחלקה Base אם CTOR של Base protected או public
ד) אי אפשר לרשת ממחלקה אם אין בה DEFAULT CTOR
ה) אף תשובה אינה נכונה

שאלה 4:

האם שורה 33 מתקמפלת ?

Drv d1;

א) מתקמפלת כי קומפיילר מספק לכל מחלקה Default CTOR
ב) מתקמפתל כי מחלקה Drv יורשת Default CTOR ממחלקה Base
ג) לא מתקמפלת כי ברגע שהוגדר איזהשהו CTOR קומפיילר כבר לא מספק Default CTOR והיא אפשר לרשת CTOR
ד) מתקמפלת כי קומפיילר יתן ערכים NULL לכל data-members ויפנה ל CTOR שמקבל פרמטרים
ה) אף תשובה אינה נכונה

שאלה 5:

האם שורה 30 מתקמפלת ?
friend class Person;

א) לא מתקמפלת כי FRIEND חייב להיות מוגדר בPUBLIC ולא בPRIVATE
ב) לא מתקמפלת כי FRIEND צריך להיות אובייקט ולא מחלקה
ג) מתקמפלת כי מותר להחריז על מחלקה כולה כ FRIEND ולא רק member-function מסוים של אותה מחלקה
ד) מחלקה PERSON יכולה להיות FRIEND רק אם יש בה לפחות פונקציה וירטואלית אחת
ה) אף תשובה אינה נכונה

שאלה 6:

האם שורה 37 מתקמפלת ?
***b2 = *b1;**

א) לא כי b1* ו b2* אובייקטום מסוג BASE ומחלקה BASE היא אבסטרקטית
ב) לא כי במחלקה BASE אין אופרטור הסמה
ג) מתקמפלת כי b1* ו b2* אובייקטום מסוג Drv ומחלקה Drv היא לא אבסטרקטית:
ד) מתקמפלת אבל תוצאה לא ידועה
ה) אף תשובה אינה נכונה

שאלה :7

למחלקה Drv מוסיפים data-members חדשים:
int& iRef; ו const int index;
איזה מימוש של Drv CTOR הוא נכון ?

א)
```
Drv (int i, char c): index(i), Base(i) {
    iRef = i;
    m_c1 = c;
    m_c2 = 'a';
}
```

ב)
```
Drv (int i, char c) : Base(i), m_c1(c), m_c2(c), iRef(i) { }
```

ג)
```
Drv (int i, char c): iRef(i), m_c1(c), m_c2(c), index(i) { }
```

ד)
```
Drv (int i, char c) : m_c1(c), m_c2(c) {
    Base(i);
}
```

ה) אף תשובה אינה נכונה

ס' מתכת: 63
ציון 6/8

שאלה 8:
נשנה שורה 33 ל: `Drv d1(77, 'x');` **מה הפלט של התוכנית?**

א)
```
a
x
in Drv::bar()
77
```

ב)
```
77
x
55
a
a
in Drv::bar()
77
```

ג)
```
55
x
77
in Drv::bar()
55
```

ד)
```
77
55
x
in Drv::bar()
77
```

ה) אף תשובה אינה נכונה

	א	ב	ג	ד	ה
שאלה 1			X		
שאלה2	X				@
שאלה 3			X		
שאלה 4			X		
שאלה 5			X		
שאלה 6			X		
שאלה 7			@		X
שאלה 8		X			

הסברים לתשובות לשאלות אמריקאיות:

- חובה לספק הסבר במידה ובחרה תשובה ה' (אף תשובה לא נכונה).

שאלה 1 : פעולה וקריאה ה-TOR) של Base אחרי ה-pop של Base ואז הכניסה לאוף ה-TOR) של Dm וקריאה pop של Dm

שאלה 2 : לא נשים במחסנית שהיה שינוי קומפילציה כי הכניסה Dm לקריאה ה-TOR) של Base אחרי ה-pop - ואנו צריכים את pop של Base.

שאלה 3 : ברור

שאלה 4 : ברור.

שאלה 5 : ברור

שאלה 6 : לא Dm היה אופטימלי, כדאי שהוא יהיה ליותר את Db ואחרי את Db ו- Da, ואז שוב שווה כי לוקח אותה כמה פעמים.

שאלה 7 : אי לזה נכון כי לא ניתן לשים ב-reference שלו ברשימת אתחול. ה- ל-Ref יש ערכים כי יש אתחול של Ref שאומנם שמקבל byvalue ונהרס בסוף ה-TOR) רק ל- Ref היה reference לאובייקט החס. כי לזה ניתן ראתה לשים const in front ל-Ref ואז שלחם הקומפיילר תהיה שגיאה לזה נכונה. אבל שוב שהוא לשים כי נכונה.

42

שאלה 8 : -

מס' מתמטי: 63

שאלון 8/8

חלק ג' –תכנות 20 נקודות

:TEMPLATES – STL

Building_t – template class. ID of buiding can be of ANY type :
Example:
IDs can be integers (1,3,5,7 2,4…) or even strings "A10" , "17Bet"…

Functionality of class: ***setBuidingID*** and ***getBuildingID***

Street_t – template class; is a container of buildings.
Example:
ID of street can be of ANY type : strings "Herzl", or even integers as in New York – 42, 20..
Functionality of class:
setStreetID , ***getStreetID***, ***addBuilding2Street, getBuilding***

City_t - template class; is container of streets
Example:
ID of City can be of ANY type : strings "Herzl", or any another type
Functionality of class:
setCityID , ***getCityID***, ***addStreet, getStreet***

1. נא לספק תכנון של המחלקות
2. אין צורך לממש פונקציות
3. יש להשתמש בSTL קונטיינרים כ data-members ופרמטרים של הפונקציות

```
template <class T> class Building_t {
public:
    virtual void setBuildingID(const T& pi_id);
    virtual T getBuildingID() const;
    ...
};
```

מדוע לקבל by ref
כדי לחסוך קריאת CTOR
ולהעדיף by value
כדי שלא ישני את
ה-id המשתנה המקורי
של המחלקה.

הערה נוספת: כיון שבמחלקת Street_t מאחסנים את Building_t ב-STL container, נצטרך לממש אותו: default CTOR (כדי להשתמש ב-STL), copy CTOR, operator=, operator<, operator==. כיון שהוא ממש לא מסובך.

כמו כן, נשים לב ש-DTOR ו-operator= שנוצרו אוטומטית ע"י הקומפיילר אינם מתאימים כיון שהם יעתיקו את ה-data member m_building_id שהוא protected:

```
T m_building_id;
```

על הלוח רשמתי שניתן לממש את זה בצורה פשוטה, כיון שבמקרה כזה אין צורך בהקצאת זיכרון דינמית, ולכן אין צורך ב-DTOR, ואין בעיה להשתמש ב-operator= שנוצר אוטומטית. המשך מופיע בעמ' הבא

```
template <class S, class T> class Street_t {
public:
    virtual void set_Street_ID(const S& pi_id);
    virtual S get_Street_ID() const;

    virtual void add_Building_2_Street(Building_t<T>& pi_building);
    virtual Building_t<T>& get_Building(const T& pi_building_id) const;
};
```

```
    vector<Building_t<T>> m_buildings;
    S m_street_id;
};
```

```
template <class Q, class S, class T> class City_t {
public:
    virtual void set_City_ID(const Q& pi_id);
    virtual Q get_City_ID() const;

    virtual void add_Street(Street_t<S,T>& pi_street);
    virtual Street_t<S,T>& get_Street(const S& pi_street_id) const;
};
```

```
    Street_t<S,T> m_streets;
    Q m_city_id;
};
```

20