

*טיפוסי נתונים מופשטים
ומחלקות*

אוהד ברזילי
אוניברסיטת תל אביב

1

*הוכחת תכונות של טיפוס נתונים
מופשט*

על נכונות, עקביות ושלמות
מספקת...

2

תזכורת: המחסנית כטיפוס נתונים מופשט

TYPES

- $STACK \langle T \rangle$

FUNCTIONS

- $put: STACK \langle T \rangle \times T \rightarrow STACK \langle T \rangle$
- $remove: STACK \langle T \rangle \rightarrow STACK \langle T \rangle$
- $item: STACK \langle T \rangle \rightarrow T$
- $empty: STACK \langle T \rangle \rightarrow BOOLEAN$
- $new: \rightarrow STACK \langle T \rangle$

המחסנית כטיפוס נתונים מופשט (2)

AXIOMS

For any $x: T, s: STACK [T]$

- A1 • $item (put (s, x)) = x$
- A2 • $remove (put (s, x)) = s$
- A3 • $empty (new)$
- A4 • $not\ empty (put (s, x))$

PRECONDITIONS

- @pre: $not\ empty (s)$
 $remove (STACK \langle T \rangle, s)$
- @pre: $not\ empty (s)$
 $item (STACK \langle T \rangle, s)$

תכונות הגדרת ה ADT

- **נכונות** (תקינות, correctness) של ביטוי
- **עקביות** (consistency) ההגדרה
- **שלמות מספקת** (sufficient completeness) של ההגדרה
- נוכיח תכונות אלו באינדוקציה על אורך הביטוי

משקל של מחסנית

- הגדרה: פונקצית המשקל, w , של מחסנית מוגדרת כהלכה ללא שימוש ב `empty` או `item` מוגדר באינדוקציה:
 - W1: $w(\text{new}()) = 0$
 - W2: $w(\text{put}(s,x)) = w(s) + 1$
 - W3: $w(\text{remove}(s)) = w(s) - 1$

טענות עזר

- **טענה (עקביות המשקל):** ביטוי מחסנית בנוי כהלכה e , ללא `item` או `empty`, הוא תקין אם ורק אם המשקל שלו אי שלילי וכל תת ביטוי של e הוא (בריקורסיה) תקין
- **טענה (משקל אפס):** יהי e ביטוי מחסנית בנוי כהלכה ללא `item` או `empty`, אזי `empty(e)` הוא `TRUE` אם ורק אם $w(e)=0$

הוכחה באינדוקציה – מקרה בסיס

- `new` הוא תקין שכן אין לא תת ביטויים ואין לו תנאי מקדים
- `empty(new) = TRUE` לפי אקסיומה A3

צעד האינדוקציה - put

- יהי $e = \text{put}(s, x)$
- ל put אין תנאי מקדים ולכן e תקין $\Leftrightarrow s$ תקין
- \Leftrightarrow ל s ולכל תת הביטויים של s יש משקל אי שלילי \Leftrightarrow ל- e ולכל תת הביטויים של e יש משקל אי שלילי
- S תקין $\Leftarrow w(s) \geq 0 \Leftarrow w(e) > 0$
- $\text{!empty}(e)$ לפי אקסיומה A4

צעד האינדוקציה – remove (עקביות המשקל)

- יהי $e = \text{remove}(s)$
- e תקין $\Leftrightarrow s$ וכל תת הביטויים שלו תקינים ומתקיים $\text{!empty}(s) \Leftrightarrow w(s) > 0$ וכל תת הביטויים של e תקינים $\Leftrightarrow w(e) \geq 0$ וכל תת הביטויים של e תקינים

צעד האינדוקציה – remove (משקל אפס)

- יהי $e = \text{remove}(s)$
 - $w(s) > 0$ אזי ל- s (וגם ל- e) מופע של put
 - המופע החיצוני ביותר של put עטוף ב- remove . נפעיל עליו את $A2$:
 $\text{remove}(\text{put}(s, x)) = s$
- אנו מקבלים ביטוי e קצר יותר עם אותו המשקל, שעבורו תכונת משקל האפס נובעת לפי הנחת האינדוקציה

כלל ההצגה הראשית (canonical reduction)

- לכל ביטוי מחסנית בנוי כהלכה ותקין אשר אינו מכיל item ו- empty יש ייצוג שקול (קנוני) אשר אינו מכיל remove (כלומר מכיל רק put ו- new)
- תכונה זו מתקבלת ע"י הפעלה חוזרת של אקסיומה $A2$

הוכחת שלמות מספקת (ועקביות) מקרה הבסיס

- ביטוי מחסנית חייב להיות `new`
- `empty(new)` תקין (וחיובי) לפי A3
- `item(new)` לא תקין מכיוון שהתנאי המקדים `!empty(new)` לא מתקיים

הוכחת שלמות מספקת (ועקביות) צעד האינדוקציה

- לפי כלל ההצגה הראשית אפשר באינדוקציה לייצג כל מחסנית ע"י `new` (מקרה הבסיס) או `put(s',x)`
- `empty(s)` הוא תקין ושילולי לפי A4
- `item(s)` הוא תקין (מכיוון שמתקיים `!empty(s)`) וערכו הוא `x` לפי A1

מחלקות בשפת C++

על הרשאות גישה, עצמים
מוכלים ועצמים מוצבעים...

15

CPoint

```
#include <iostream>

class CPoint
{
    int m_x; /** X coordinate */
    int m_y; /** Y coordinate */

public:
    /** Set the coordinates to zero */
    void Init()
    {
        m_x = 0;
        m_y = 0;
    }
}
```

16

CPoint

```
/** Set the coordinates of the point */  
void Set(int ax, int ay)  
{  
    m_x = ax;  
    m_y = ay;  
}  
  
/** Print the point's coordinates */  
void Print()  
{  
    cout<< "x = " << m_x <<  
        " , y = " << m_y << endl;  
}  
};
```

Using CPoint

```
int main()  
{  
    CPoint p1, p2;  
    p1.Init();  
    p2.Set(4,6);  
    p1.Print();  
    p2.Print();  
}
```

CPoint הדור הבא

```
/** Set the coordinates of the point */
bool Set(int ax, int ay)
{
    if (InRange(ax) && InRange(ay)) {
        m_x = ax;
        m_y = ay;
        return true;
    }
    else
        return false;
}
int GetX() { return m_x; }
int GetY() { return m_y; }

private:
/** Check that a given value is in range */
bool InRange(int av) { return (av >= 0 && av <= 100); }
};
```

תכנות מונחה עצמים בשפת C++
אוניברסיטת תל אביב

19

CPoint הדור הבא

```
int main()
{
    CPoint p1, p2;
    p1.m_x = 12;           // ERROR (private member)
    p1.m_y = 10;          // ERROR (private member)
    p2.InRange(34);       // ERROR (private member)

    if (p1.Set(20,35))
    {
        p1.Print();
        p2.Set(p1.GetX(), p1.GetY()); // same as p2=p1
        p2.Print();
    }
}
```

תכנות מונחה עצמים בשפת C++
אוניברסיטת תל אביב

20

CRectangle

```
class CRectangle
{
    // private (default)
    CPoint m_TopLeft;           /** The top left corner */
    CPoint m_BottomRight;      /** The bottom right corner */

public:
    /** set all the coordinates of the rectangle */
    bool Set(int atlx, int atly, int abrx, int abry)
    {
        if (m_TopLeft.Set(atlx, atly) &&
            m_BottomRight.Set(abrx, abry))
            return true;
        else
            return false;
    }
}
```

CRectangle

```
/** set coordinates of rectangle (function overloading) */
void Set(const CPoint& ptl, const CPoint& pbr)
{
    m_TopLeft = ptl;
    m_BottomRight = pbr;
}

void Print()
{
    cout << "Top Left Corner: ";
    m_TopLeft.Print();
    cout << "Bottom right Corner: ";
    m_BottomRight.Print();
}
};
```

Using CRectangle

```
int main()
{
    CRectangle rect;
    rect.Set(10,10,30,24); // using 1st method
    rect.Print();

    CPoint p1, p2; // using 2nd method
    p1.Set(20,45);
    p2.Set(50,70);
    rect.Set(p1,p2);
    rect.Print();
}
```

CColoredRectangle

```
class CColoredRectangle
{
    CRectangle m_rect;
    enum Color { RED=0 , GREEN=1 , BLUE=2 } m_Color;

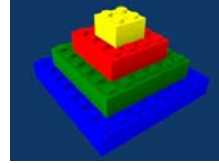
public:
    /** set the colored rectangle using a rectangle */
    void Set(const CRectangle & rec, Color col=RED)
    {
        m_rect = rec;
        m_Color = col;
    }

    void Print()
    {
        m_rect.Print();
        cout << "color is " << m_Color << endl;
    }
};
```

Using CColoredRectangle

```
int main()
{
    CRectangle rect;
    rect.Set(10,10,30,24);

    CColoredRectangle c_rect;
    c_rect.Set(rect);
    c_rect.Print();
}
```



דיון

- האם יש הבדל בין הקשרים הבאים:
 - מלבן - נקודה (המלבן מכיל שדות מטיפוס CPoint)
 - מלבן - מלבן צבעוני (מלבן צבעוני מכיל שדה מטיפוס מלבן)
- המבנה הרקורסיבי של נקודה, מלבן ומלבן צבעוני הוא דוגמא לעקרון הפשטת הקופסא השחורה ולעקרון ההכמסה (encapsulation)

מצביעים ל CPoint

```
int main()
{
    CPoint *pP1, *pP2;
    CPoint P3, P4, P5;
    P3.Set(3,11);           // OK
    pP1->Set(45,34);       // RUNTIME ERROR
    pP1 = new CPoint;
    pP1->Set(45,34);       // OK NOW
    P4 = P3;               // Copy all data members of P1 to P2
    pP2 = &P4;             // pP2 now points to P4;
    pP2->Print();          // OK - Will print the data of P4;
    P5 = *pP1;             // OK - copy the data members of the
                          // object pointed by pP1

    P5.Print();
    // And don't forget to free memory
    delete pP1;
    delete pP2;           // NO WAY - Remember - Don't free
                          // memory you did not allocate
}
```

C++ תכנות מונחה עצמים בשפת
אוניברסיטת תל אביב

27

הפנייה ל CPoint

```
int main()
{
    CPoint P1;
    CPoint& P2 = P1; // now P1 and P2 reference the
                   // same object

    CPoint& P3;     // Error! Reference must be
                   // initialized

    P1.Set(4,5);
    P2.Print();

    P2.Set(5,4);
    P1.Print();
}
```

C++ תכנות מונחה עצמים בשפת
אוניברסיטת תל אביב

28

מורים וקורסים

```
class CTeacher; // declaration of class CTeacher
class CCourse
{
    char    *name; /** name of the course */
    long    serial;
    CTeacher *pCourseTeacher;
    ...
public:
    /** Get the pointer to the teacher of this course */
    CTeacher *GetTeacher() { return pCourseTeacher; }

    /** Assign a new teacher to this course */
    bool AssignTeacher(CTeacher* pTeacher)
    {
        if (pCourseTeacher != NULL)
            return false;
        pCourseTeacher = pTeacher;
        return true;
    }
};
```

מורים וקורסים

```
class CTeacher
{
    char    *name; /** name of the teacher */
    int     vetek;
    CCourse *pCourses[10]; /** all courses of
                           teacher */
    ...
public:
    /** Add a new course to this teacher */
    bool AddCourse(CCourse *pNewCourse)
    {
        ...
    }
};
```

דיון

- למה בדוגמת המורים והקורסים השתמשנו במצביעים ואילו בדוגמת המלבן והנקודות השתמשנו במחלקות עצמן?

שימוש ב this

```
// declaration
class CPoint;
void PrintNice(CPoint *p); // Global function

class CPoint {
    ...
    Print() { PrintNice(this); }
    CPoint& Inc()
    {
        m_x++;
        m_y++;
        return *this;
    }
};

void PrintNice(CPoint *p) {
    cout << "x = " << p->GetX() <<
        ", y = " << p->GetY() << endl;
}
```

- קריאה לפונקציות גלובליות
- שרשור של פעולות

שימוש ב this

```
int main()
{
    CPoint P1;
    P1.Set(10,20);
    P1.Print();
    PrintNice(&P1);
    P1.Inc();
    P1.Inc().Inc().Inc();
}
```

