A Familiar yet Vague Name: “Abstract Data Type”

C++ Object Oriented Programming

Pei-yih Ting

NTOU CS

Abstract Data Type

抽象資料型態

- **Abstract**?!
  - Disassociated from any specific instance 抽象的, 不易懂的
  - Expressing a quality apart from an object 抽象化 (理論化)
  - Having only intrinsic form with little attempt at pictorial representation or narrative content 摘要
- **Data type**?

  characteristics of a set of data,

  template for instances of data storage

  specifies: { format, ranges, memory resources }

Abstract Data Type (cont’d)

- See what people on Internet said

  何謂ADT(Abstract data type)
  我一直搞不懂ADT是啥?
  抽象資料型態(ADT)
  我知道是一個自訂的資料型態,
  但是卻似懂非懂,
  可以幫忙解釋一下嗎?
  感謝...

Any better?!

簡單的說陣列 (array) 就是一種抽象的觀念,
但是你做出了 int array[10]; 這樣的實踐, 就是抽象觀念的實作...

Abstract Data Type (cont’d)

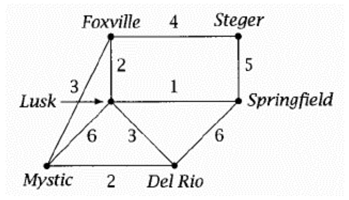
- http://en.wikipedia.org/wiki/Abstract_data_type
- In computing, an abstract data type (ADT) is a specification of **a set of data** and **the set of operations** that can be performed on the data.
- e.g. container, deque, list, map, multimap, multiset, priority queue, queue, set, stack, string, tree
- Such a data type is ***abstract*** in the sense that <u>it is independent of various concrete implementations</u>.
  - Question: Will they still be abstract without the set of operations (only the set of data)??

Abstract Data Type (cont’d)

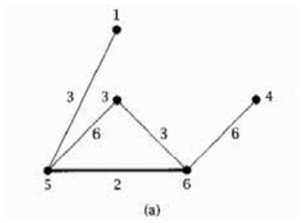
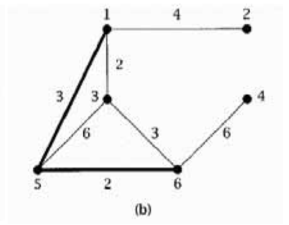
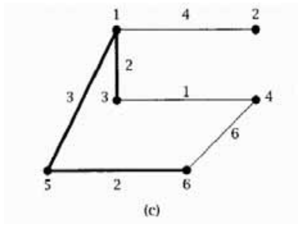
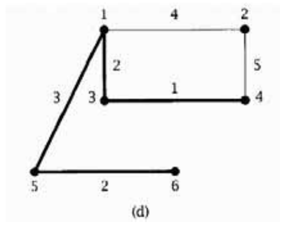
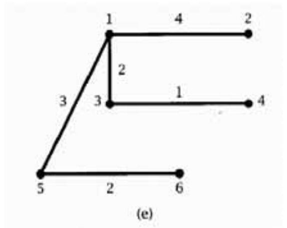
- Are you really satisfying with this definition???
  - “Data type” is an easy idea: the attributes
  - It looks like that “data type” itself could also be independent of various implementations.
  - Why is the additional “operations” related to the keyword “abstract”???

Example: Prim’s MST

- In JohnsonBaugh’s “Algorithms”



Minimum Spanning Tree

Example: Prim’s MST (cont’)

```
prim(adj, start, parent) {
   n = adj.last
   for i = 1 to n
      key[i] = ∝
   key[start] = 0
   parent[start] = 0
   h.init(key, n)
   for i = 1 to n {
      v = h.del()
      ref = adj[v]
      while (ref != null) {
         w = ref.ver
         if (h.isin(w) &&
             ref.weight < h.keyval(w)) {
             parent[w] = v
             h.decrease(w, ref.weight)
         }
         ref = ref.next
      }
   }
}
```

h is an abstract data type that supports the following operations
h.**init**(key, n): initializes h to the values in key
h.**del**(): deletes the item in h with the smallest weight and returns the vertex
h.**isin**(w): returns true if vertex w is in h
h.**keyval**(w): returns the weight corresponding to vertex w
h.**decrease**(w, new_weight): changes the weight of w to new_weight (smaller)

Abstract Painting

- Picasso

Miro - Angel

抽象畫 - 非寫實畫風

Abstract

- Mathematic formula: Central Limit Theorem, Stirling formula, Fourier Transform, …
- Physic formula: wave equation, …

Quite often is the case that you cannot see what these formula mean because they are deprived of from their original application environments.
Thus, you say that these formula are quite abstract.

Abstraction

- **Abstraction**: the process or result of generalization by reducing the information content of a concept or an observable phenomenon
  - A method to find general form of an idea
  - A method to find a unified explanation
  - A method to simplify the complex exteriors.
  - 抽象化 – 單純化 – 簡化
  - ex. 鳥可以飛, 飛機可以飛, 蚊子可以飛 ➔ 有翅膀的
    but 鴕鳥, 肉雞…

  需要描述翅膀怎麼用才能飛 – 需要有操作型定義
  一個資料結構真正代表的意義 – 必需用動作來描述

Data vs. Operation

- 茶杯 ..... pure data

- Data can be used for any imaginable purpose.
- You want your data storage to be specific. You name its “operations”
  - How do you use this data?
  - For what do you use it?

Back to ADT

- abstract data type (ADT):
  is a specification of
  - **a set of data** and
  - **the set of operations** performed on the data.

- It is independent of various implementations
- It provides specific descriptions of the functionalities of a piece of data in terms of operations.

The C syntax: x.y vs. x.z()

- In C, how do you capture the idea of

  h.key and **h.decrease(w, weight)**
- Are these two syntactically correct in C?
- Yes.
- decrease is called a “function pointer”
- It is a piece of data (attribute), and at the same time, you can invoke a function via this data.
  - e.g.

```
void fun(int x)
{
    …
}
```

```
void (*fp)(int);
 …
fp = fun;
(*fp)(5);  /* calling fun(5) */
```

```
// cl testfp.c
#include <stdio.h>

struct MyStruct
{
    int x;
    int (*fp)(int);
};

int isOdd(int data);

void main()
{
    struct MyStruct obj = {123, isOdd};
    int data;
    int (*myfp)(int) = isOdd;

    printf("Please input an integer: ");
    scanf("%d", &data);
    printf("%d\n", obj.fp(data));
    printf("%d\n", (*obj.fp)(data));
    printf("%d\n", myfp(data));
    printf("%d\n", (*myfp)(data));
    printf("%d\n", isOdd(data));
}

int isOdd(int data)
{
    printf(" calling isOdd() ");
    if (data%2 == 1)
        return 1;
    else
        return 0;
}
```