COMP1511 17s2

— Lecture 16 —

Using Abstraction

~~Andrew Bennett~~

~~<andrew.bennett@unsw.edu.au>~~

Jashank Jeremy

<jashank.jeremy@unsw.edu.au>

review: abstract data types

applications of ADTs

Don’t panic!

assignment 1 **reflection**

due Wed 20 Sep, 23:59:59 (Wednesday, week 9)

MandelbrArt

… The MandelbrArt Gallery

closes Wed 20 Sep, 23:59:59 (Wednesday, week 9)

vote on the most beautiful image soon!

Abstract Data Types

separating the **interface** from the **implementation**

Abstraction with ADTs

implementation vs **interface**

interface: opaque values; details hidden from user

implementation: `struct` and function definitions

interface is a *well-defined boundary*...

implementation shouldn’t trust users;

users shouldn’t trust implementation

Abstraction with ADTs

implementation vs interface

interface: opaque values; details hidden from user

implementation: `struct` and function definitions

```
// in Complex.h:
typedef struct _complex *Complex;

Complex newComplex (double re, double im);
void destroyComplex (Complex c);
double complexRe (Complex c);
double complexIm (Complex c);
double complexMod (Complex c);
double complexArg (Complex c);
```

how does this ADT store values? (x, y)? (r, θ)?

we don't know, and **don't need to know**

Abstraction with ADTs

implementation vs interface

interface: opaque values; details hidden from user

implementation: `struct` and function definitions

```
// in Complex.c:
typedef struct _complex {
    double re;
    double im;
} complex;

complex *newComplex (double re, double im) { /*
... */ }
void destroyComplex (void) { /* ... */ }
double complexRe (complex *c)  { /* ... */ }
double complexIm (complex *c)  { /* ... */ }
double complexMod (complex *c) { /* ... */ }
double complexArg (complex *c) { /* ... */ }
```

Why?

why bother with all this effort?
what do ADTs let us do
(that we couldn't already)?

"... the purpose of abstraction is **not to be vague**,
but to **create a new semantic level** in which
one can **be absolutely precise**."

— from *The Humble Programmer* by E. W. Dijkstra (EWD 340)

we can use ADTs to
build a new level of abstraction

a concrete stack

((demo: stack.h))

((demo: stack.c))

an abstract stack

((demo: Stack.h))

((demo: Stack.c))

((demo: testStack.c))

enum

```
typedef enum {
    CONSTANT-NAME = VALUE,
    CONSTANT-NAME[ = VALUE],
} TYPE-NAME;
```

like `struct` , a new type of type
defines symbolic names for values like `#define` ,
and constrains that type to those values

A Card ADT

playing cards, for the game of

Final Card-Down...