Abstract data types (ADT's)

Data Structures

- data – factual information
- structure – arrangement or relationship of elements
    - The New Merriam-Webster Pocket Dictionary

Hierarchy of Data

Data Type

- A data type is defined by two properties:
  - A Domain: set of values that belong to that type.
  - A Set of Operations: defines the behavior of that type.

- Example: int
  - What is the domain?
  - What are the operations?

Primitive (Atomic) C Data Types

Defined by the C language specification.

- Integer types: short, int, long
  - These can be preceded by unsigned
  - What is the size of int?
  - sizeof (int)
- Floating point types: float, double, long double
- Text types: char
  - What about string?
- What about Boolean?

Goal

- Build a useful data structure
- Test the data structure
- Hand that data structure off to a customer
  - future you
  - teammate
  - paying customer
  - your professor
- Profit / pass the class!

Data Structures

- A data structure can be thought of as a data type with values that:
  - Can be broken up into a set of component elements where each element is either atomic or another data structure
  - Include a set of relationships (structure) involving the component elements

Abstract Data Types (ADTs)

- An Abstract Data Type is defined in terms of its behavior rather than its representation

- An ADT has two qualities:
  - Irrelevant details are suppressed (hidden)
  - The data type being abstracted is isolated

- When defining ADTs you need:
  - The Domain
  - The Operations

Data Structure vs. ADT

- An ADT is defined by its behavior from the point of the ***user***

- A data structure is a concrete implementation of an ADT from the point of the ***implementer***

INTEGER ADT

Integer ADT

- Let us consider the specification for the integer ADT as follows:
- ADT: Integer
- Domain: All whole numbers i where

  INT_MIN <= i <= INT_MAX

  <limits.h>

Integer ADT Specification

- Operations: Given i is an integer and f & g are expressions whose result is an integer, we define the following operations for C:

Operator	Results
Unary +	+f is the same as f
Unary -	-f changes the sign of f
Assignment =	i = f assigns the integer value of f to i
Binary +	f + g is the addition of two integer values

STRING ADT

String ADT

- Integer is an atomic ADT
- How might we specify a structured data type such as a String?
- Before specifying the String ADT, we need to answer certain questions

String ADT Questions

- What are the domain of possible values
- What operations exist?

Language independent

Language specific

- What type are the component elements?
- What structure does the type have?

String ADT Specification

- Elements: Type char excluding the null terminating character.
- Structure: Characters are arranged linearly.
- Domain: All combinations of strings of length 0 to the max string length that can be formed from the character set.

String ADT Specification

- Operations
  - function strLength (s)
    - results: returns the number of characters in the string s
  - function strEqual (s1, s2)
    - results: returns true iff strLength (s1) equals strLength (s2) and the ith character of s1 and s2 are equal for all i where 1 <= i <= strLength (s1) *
  - function strConcat (s1, s2)
    - results: string s2 is concatenated on the end of string s1; if the result exceeds the max string length, the characters are dropped

*Why 1 and not 0?

String ADT Specification

- Operations Continued
  - function strAppend (s, ch)
    - requires: strLength (s) < max string length
    - results: ch is added to the end of s increasing the length by 1
  - function strReverse (s)
    - results: the characters of s are reversed “abc” is “cba”
  - function strClear (s)
    - results: the string s is made empty
  - function strCopy (s1, s2)
    - results: string s2 is copied into string s1

ADT Implementation

- Now that the String ADT has been specified, we can focus on the best implementation choice.
- Before writing the code for each of the functions, we need to decide how we are going to represent a string.
- Code defensively! Your Data Structure should never crash the user's program.

String Representation

```
#define MAX_STR_LEN 256
typedef struct
{
  int length;
  char data[MAX_STR_LEN];
} String;
```

Problem

- For the given String representation, implement each of the following functions in C:
  - strLength
  - strCopy
  - strConcat