Lecture 2 – Introduction to Abstract Data Types

CST238 – Intro to Data Structures
YoungJoon Byun
ITCD

Lecture Objectives

- After completion of this lecture, you will be able to

Chapter 2: Introduction to Abstract Data Types

- **2.1 A first look at ADTs and Implementations**
- 2.2 C++'s Simple Data Types
- 2.3 Programmer-Defined Data Types
- 2.4 Pointers

Abstract Data Type (ADT)

- For a programming task, a programmer must identify
  - The collection of data items
  - Basic operations to be performed on them
- **A**bstract **D**ata **T**ype (ADT) takes data items and operations on them together.

Chapter 2: Introduction to Abstract Data Types

- 2.1 A first look at ADTs and Implementations
- **2.2 C++'s Simple Data Types**
- 2.3 Programmer-Defined Data Types
- 2.4 Pointers

A Sample C++ Program

```
1.  /*
2.   * Title: average.cpp
3.   * Abstract: This program computes the average of
4.   *           three exam scores.
5.   * Author: Dr. Byun
6.   * ID: XXXX
7.   * Date: 08/24/09
8.   */
9.  #include <iostream>
10. using namespace std;
11. int main ()
12. {
13.   double score1, score2, score3, average;
14.   cout << "Enter three scores: ";
15.   cin >> score1 >> score2 >> score3;
16.   average = (score1 + score2 + score3)/3;
17.   cout << "The average is " << average << endl;
18.   return 0;
19. }
```

C++ Data Types – Overview

- char, bool, short, int, long
- float, double
- pointer, reference
- array, string,
- struct, class
- …

Integer Data Types

- Signed integers
  - **`short, int, long`**
  - They are represented in two's complement notation.
- Unsigned integers
  - **`unsigned short, unsigned, unsigned long`**
  - They are useful when you need to store positive numbers such as population and age.
  - Example: **`unsigned short age = 21;`**

Two's Complement Representation

- For a nonnegative number *n*
  - Use ordinary base-two representation with leading (sign) bit 0
- For a negative number *n*

  1) Find base-2 representation of |**n**|

  2) Complement each bit.

  3) Add 1

Two's Complement - Example

sizeof Operator

- This operator determines the size of a variable or a data type.
- Example

```
short  shortVar = 100;
int  intVar;
cout << “shortVar has " << sizeof(shortVar) << " bytes.\n";
cout << “intVar has " << sizeof(int) << " bytes.\n";
cout << “A long type has " << sizeof(long) << " bytes.\n";
```


A serious problem in Integer Representation

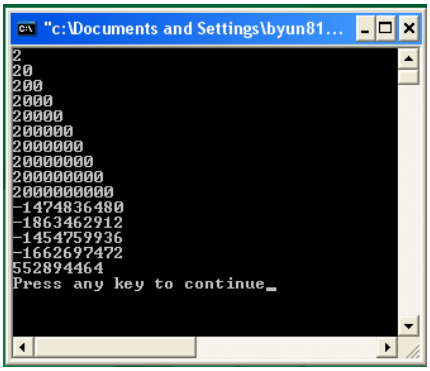
Integer Overflow Demonstration

```
1.  /* Fig.2.1 of Textbook Sample Code */
2.  #include <iostream>
3.  using namespace std;

4.  int main()
5.  {
6.     int number = 2;

7.     for (int i = 1; i <= 15; i++)
8.     {
9.        cout << number << endl;
10.       number *= 10;
11.    }
12. }
```


Execution Result



Integer Overflow Demonstration (2)

```
1.  /* Fig.2.2 of Textbook Sample Code */
2.  #include <iostream>
3.  #include <climits>
4.  using namespace std;

5.  int main()
6.  {
7.     int number = INT_MAX - 3;

8.     for (int i = 1; i <= 7; i++)
9.     {
10.       cout << number << endl;
11.       number++;
12.    }
13. }
```

Data Types for Real Numbers

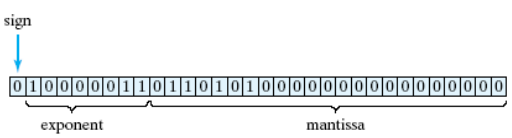
- **`float`**, **`double`**, or **`long double`** in C++
  - A real number can contain a decimal point.
  - e.g., 3.141592
- E notation (or Scientific notation)
  - Example: 3.67e3 means 3670.0
    3.49e-2 ➔ 0.0349
    5.89e-6 ➔ 0.00000589

Internal Representation of Real Numbers in a Computer

- A computer uses single precision (IEEE Floating-Point)
- It needs to store
  (1) sign of mantissa in leftmost bit (0 = +, 1 = –)
  (2) exponent in next 8 bits
  (exponent + 127)
  (3) bits $b_2b_3 \ldots b_{24}$ mantissa in rightmost 23 bits.
  - We don't need to not store b_1 because we know it's always 1.

Real Data Representation

- Example: **`22.625`** = **$\mathtt{10110.101_2}$**
- Floating point form:
  $\mathtt{1.0110101_2 * 2^4}$



Problems with Real Representation

- Exponent overflow and underflow
- Round off error
  - Most real numbers do *not* have terminating binary representations.
  - Example:

$0.7 = (0.10110011001100110011001100\ldots)_2$

Problems with Real Representation

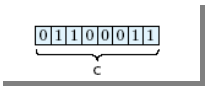
- Round off error may be compounded in a sequence of operations.
  - Real-world example: Gulf War Patriot missile guidance affected by accumulated round off
- **Be careful in comparing real numbers with == and !=.**

Example

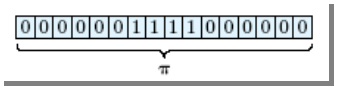
```
1. int main()
2. {
3.    for (double x = 0; x != 5.0; x += 0.1)
4.    {
5.       cout << "x = " << x << endl;
6.    }
7. }
```

Character Data Type

- 1 byte for ASCII, EBCDIC



- 2 bytes for Unicode (Java) or C++ *wide character type*



Boolean Data Type

- e.g., bool errorFlag;
  - **`errorFlag`** can have either **`false`** or **`true`**.

  errorFlag = true;

- This type could be stored in bits, usually use a byte

Chapter 2: Introduction to Abstract Data Types

- 2.1 A first look at ADTs and Implementations
- 2.2 C++'s Simple Data Types
- **2.3 Programmer-Defined Data Types**
- 2.4 Pointers

Programmer-Defined Data Type: **`typedef`**

- A mechanism to create a new data type
  - You can give a new name (= alias) to an existing data type.

  `typedef` *`OldType NewType`*`;`
- Example

  ```
  typedef double real;
  ```
  - Now both **`double`** and **`real`** can be used.

Programmer-Defined Data Type: **`enum`**

- A mechanism for creating a data type whose values are identifiers
  - Each identifier associated with unique integer
- Example

```
enum Color {RED, ORANGE, YELLO, GREEN,
            BLUE, INDIGO, VIOLET};
Color shade = BLUE;  // shade is 4;
```

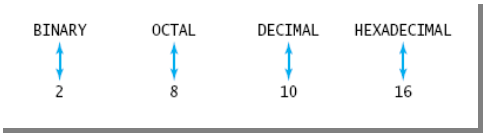
Example

```
enum Day { MONDAY, TUESDAY,
           WEDNESDAY, THURSDAY,
           FRIDAY };
Day workDay;
workDay = WEDNESDAY;
```

Using enumerations: **enum**

- It's also possible to specify explicit values to give the enumerators.

```
enum NumberBase { BINARY = 2,
                  OCTAL = 8,
                  DECIMAL = 10,
                  HEXADECIMAL = 16};
```



Summary

- C++'s Simple Data Types
  - short, int, long, float, double, char, bool, unsigned
- Programmer-Defined Data Types
  - typedef, enum
- Next Lecture
  - Pointers (Chap. 2.4)

References

- Larry Nyhoff, *ADTs, Data Structures, and Problem Solving with C++*, 2nd Edition, Prentice-Hall, 2005
- Walter Savitch, *Problem Solving with C++*, 6th Edition, Addison-Wesley, 2006
- Dr. Meng Su's Lecture Notes
  http://cs.bd.psu.edu/~mus11/122Fa06/cse122Fa06.htm