Numerical Conversions Intro to Strings in C/C++

CS 16: Solving Problems with Computers I
Lecture #8

Ziad Matni Dept. of Computer Science, UCSB

Announcements

- We are grading your midterms this week!
 - Grades will be available by Tuesday next week
- Next lab is a required pair programming assignment
 - You are required to work with a partner for Lab 5 (next week)
 - Read the description for Lab 5 when it comes out over the weekend carefully for the details

Note: Functions Calling Functions

- A function body may contain a call to another function
- The called function declaration must still appear before it is called
- Functions cannot be defined in the body of another function
 - Including inside the main() function!

```
void order (int&, int&);
void swap_values (int&, int&);

int main () {
...
    order (a, b);
...
    return 0; }

void order(int& n1, int& n2) {
    if (n1 > n2)
        swap_values(n1, n2); }

void swap_values(int& n1, int& n2) {
    int temp = n2;
    n2 = n1;
    n1 = temp; }
```

Positional Notation in Binary

11011 in base 2 positional notation is:

$$1 \times 2^{4} = 1 \times 16 = 16$$
 $+ 1 \times 2^{3} = 1 \times 8 = 8$
 $+ 0 \times 2^{2} = 1 \times 4 = 0$
 $+ 1 \times 2^{1} = 1 \times 2 = 2$
 $+ 1 \times 2^{0} = 1 \times 1 = 1$

So, **11011** in base 2 is 16 + 8 + 0 + 2 + 1 = 27 in base 10

10/26/17 Matni, CS16, Fa17

Converting Binary to Octal and Hexadecimal

(or any base that's a power of 2)

- Binary is 1 bit
- Octal is 3 bits $(2^3 = 8)$ octal is base 8
- Hexadecimal is 4 bits $(2^4 = 16)$ hex is base 16
- Use the "group the bits" technique
 - Always start from the least significant digit
 - Group every 3 bits together for bin \rightarrow oct
 - Group every 4 bits together for bin \rightarrow hex



• Take the example: 10100110

...to octal:

10100110 2 4 6

...to hexadecimal:

10100110 10 6 246 in octal

A6 in hexadecimal

Decimal 4 5 6

8

Α

В

C

D

E

Octal symbols

Hex. symbols

6

Converting Decimal to Other Bases

Algorithm for converting number in base 10 to other bases While (the **quotient** is not zero)

- 1. Divide the decimal number by the new base
- 2. Make the remainder the next digit to the left in the answer
- 3. Replace the original decimal number with the quotient
- 4. Repeat until your quotient is zero **EXAMPLE**:

Convert the decimal (base 10) number 79 into hexadecimal (base 16)

The answer is: **4F**

Converting Decimal into Binary

Convert 54 (base 10) into binary and hex:

- 54/2 = 27 R 0
- 27 / 2 = 13 R **1**
- 13 / 2 = 6 R **1**
- 6/2 = 3 R **0**
- 3 / 2 = 1 R 1
- 1/2 = 0 R 1

```
Sanity check:
110110
= 2 + 4 + 16 + 32
= 54
```

Class Exercises

Convert these binaries into decimal AND hexadecimal:

1010101010 10100001000

Ans: 682 (dec) 2AA (hex) 1288 (dec) 508 (hex)

Convert these decimals into binary:

65

333

Ans:

1000001 101001101

An Intro to Strings

Strings in C/C++

- Recall: C++ is based on C
- Originally (in C), strings were defined as an "array of characters"
 - Called C-Strings and are "legacy" data types in C++
 - Came with the library <cstring>
 - Contains lots of built-in functions that go with C-Strings
- In C++, we got a new library: <string>
- Made improvements over the old "C-String"
 - Library contains another collection of functions that work with Strings, but not C-Strings!

Why Do We Care About C-Strings??

- Their use STILL crops up in C++
 - Recall: command line arguments...

What's with the "star"???

* Is used here to indicate a pointer.

We'll discuss these at a later date.

- Recall that command-line arguments, specifically argv[x] are defined as:
- That's a classic definition of a C-String
 - So if we want to use these argv[x], we'll have to treat them in a C-String fashion...

What is a String?

Characters connected together in a sequence

P	i	k	a	c	h	u
Н	i		M	0	m	!

C strings vs. C++ strings

- C++ is meant to be backwards compatible with C
- C has one way of dealing with strings, while C++ has another
- C++'s use is much easier and safer with memory allocation
 - This is what you've learned so far with <string>
 - Let's briefly look at the other (older) way with C-strings...

What's a C++ Programmer to Do?!

- Be aware of 3 types of variables that deal with a bunch of connected characters...
- An "ordinary" array of characters (we'll deal with arrays later...)
 - Like any other array: no special properties that other arrays do not have
- A C-string (that's the "old" style)
 - An array of characters that's ALWAYS terminated by the null character '\0'
 - The null character has an ASCII code of **0**.
 - Library for dealing with these types: <cstring>
- A C++ string object
 - An instance of a "class" data type used as a "black box" by programmers
 - Library for dealing with these types: <string>

The C String

- The null-character terminates the actual string, but not the array necessarily
- Example: a C-string stores "Hi Mom!" in a character array of size 10
 - The characters of the word "Hi Mom!" will be in positions with indices 0 to 6
 - There will be a null character at index 7, and the locations with indices 8 to 9 will
 contain some undefined value that we won't care about
 - It's the null character at index 7 that makes this otherwise ordinary character array a C-string.

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
Н	i		M	0	m	!	\0	??	??

The C++ String

- There is NO null-character to worry about!
- There are no array-related set-ups you have to worry about!
- There are some "cool" built-in functions
 - More on those later...

Declaring a String in C++

You have to include the correct library module with:

```
#include <string>
```

Declare them (and initialize them) with:

```
string MyString=""; // Note the use of double-quotes!
```

• Since strings are made up of characters, you can **index** individual characters in strings (starting at position 0), using the square brackets []:

```
If MyString = "Hello!"
Then MyString[0] = 'H', MyString[1] = 'e', etc...
```

REMEMBER: " vs '

- Double quotes are used exclusively for strings
 - —Both C and C++ strings
- Single quotes are used **exclusively** for characters

Streams and Basic File I/O

- Files for I/O are the same type of files used to store programs
- A stream is a flow of data
- Input stream: Data flows into the program
 - E.g. cin
- Output stream: Data flows out of the program
 - E.g. cout
- You can also use data streams with files

Why Use Files?

- Files allow you to store data permanently!
- Data output to a file lasts after the program ends
 - You can usually view them without the need of a C++ program
- An input file can be used over and over
 - No typing of data again and again for testing
- Create or read files at your convenience
- Files allow you to deal with larger data sets

YOUR TO-DOs

☐ Lab 4 due Fri. 10/27

☐ HW5 released today, due in 1 week.

☐ Lab 5 will be released on Friday for Monday labs.

☐ Visit Prof's and TAs' office hours if you need help!

☐ Floss!

