Input / Output

Arnaud Malapert, Gilles Menez, Marie Pelleau

Master Informatique, Université Côte d'Azur

Methods of Input/Output

There are several approaches to reading in the text input.

- Repeatedly get single characters
  - (perhaps using a library function like getchar);
- Repeatedly get strings and break them down into single characters.
  - (perhaps using a library function like scanf).
- Read the entire line as a string, and then parsing it by accessing characters in the string.
  - (perhaps using a library function like gets).

Perhaps more modern ways using streams are easier, perhaps not.

Make the Best from your Language

Basic Data Types

Selecting the right data structure makes a trememdous difference in the organization and complexity of a given program. Be aware of your basic structured data types (arrays, records, multidimensional arrays, enumerated types) and what they are used for.

Library Functions

Character Codes

Character codes

Character codes are mappings between numbers and the symbols which make up a particular alphabet.

ASCII

The American Standard Code for Information Interchange (ASCII) is a single-byte character code where $2^7 = 128$ characters are specified. Bytes are eight-bit entities; so that means the highest-order bit is left as zero.

Properties of ASCII

- Both the uppercase and lowercase letters and the numerical digits appear sequentially. Thus we can iterate through all the letters/digits simply by looping from the value of the first symbol (say, 'a') to value of the last symbol (say, 'z').
- We can convert a character (say, 'I') to its rank in the collating sequence (eighth, if 'A' is the zeroth character) simply by subtracting off the first symbol ('A').
- We can convert (say 'C') from uppercase to lowercase by adding the difference of the upper and lowercase starting character ('C'-'A'+'a')
- The character code tells us what will happen when naively sorting text files. Sorting alphabetically means sorting by character code. Using a different collating sequence requires more complicated comparison functions.
- Non-printable character codes for new-line (10) and carriage return (13) are designed to delimit the end of text lines. Inconsistent use of these codes is one of the pains in moving text files between UNIX and

Unicode

More modern international character code designs such as Unicode use two or even three bytes per symbol, and can represent virtually any symbol in every language on earth.

- Older languages, like Pascal, C, and C++, view the char type as virtually synonymous with 8-bit entities.
- However, good old ASCII remains alive embedded in Unicode. Java, on the other hand, was designed to support Unicode, so characters are 16-bit entities.

Representing Strings

Strings are sequences of characters, where order clearly matters. It is important to be aware of how your favorite programming language represents strings, because there are several different possibilities:

Null-terminated Arrays C/C++ treats strings as arrays of characters. The string ends the instant it hits the null character '0', i.e., zero ASCII. Failing to end your string explicitly with a null typically extends it by a bunch of unprintable characters

Array Plus Length Another scheme uses the first array location to store the length of the string, thus avoiding the need for any terminating null character. Presumably this is what Java implementations do internally.

Linked Lists of Characters Text strings can be represented using linked lists, but this is typically avoided because of the high space-overhead associated with having a several-byte pointer for each single byte character.

Which String Representation?

The underlying string representation can have a big impact on which operations are easily or efficiently supported. Compare each of these three data structures with respect to the following properties:

- Which uses the least amount of space? On what sized strings?
- Which constrains the content of the strings which can possibly be represented?
- Which allow constant-time access to the i-th character?
- Which allow efficient checks that the i-th character is in fact within the string, thus avoiding out-of-bounds errors?
- Which allow efficient deletion or insertion of new characters at the i-th position?

I/O Bounds

Definition

The I/O bound refers to a condition in which the time it takes to complete a computation is determined principally by the period spent waiting for input/output operations to be completed.

Van Neuman Bottleneck

Since data must be moved between the CPU and memory along a bus which has a limited data transfer rate, the data bandwidth between the CPU and memory tends to limit the overall speed of computation.

Why is it undesirable?

The CPU must stall its operation while waiting for data to be loaded or unloaded from main memory or secondary storage.

TEST - Life, the Universe, and Everything

Quickstart

Simply modify a solution of the ECHO Problem in your preferred language.

ECHO Problem

The ECHO repository contains templates for some popular languages. You can contribute if your favorite language is missing.

HELLOKIT - Hello Kitty

Algorithm 1: RectangularPattern

Data: a word w and a positive integer n.
for $i \in |w| \ldots 1$ **do**
    print the last i letters of w;
    print $n-1$ times the word w;
    print the first $i-1$ letters of w;

With String Arithmetic in Python

```
>>> x = 'Love'
>>> x[-1] + x[0:len(x)-1]
'eLov'
```

With Character Buffer in C

```
char x[] = "Love";
printf("%s%.2s", &x[2], x);
// prints 'veLo'
```

706 - LC-Display

Algorithm 2: LC-Display

Data: an integer n and a size w.
foreach $r \in [1, 2s + 3]$ **do**
    `/* In practice, split the loop into multiple ones */`
    foreach digit d **of** n **do**
        `print` the r-th row with size s of the digit d;
        `print` a space if d is not the last digit of n
    `print` a newline

Encoding a Digit of size 1

```
const char lookupTable[10][5][3] = {
  { " _ ",
    "| |",
    "   ",
    "| |",
    " _ "},
    ...};
```