
Computer Programming: Skills & Concepts (CP1)

Pattern matching with arrays; Bitwise operators

22 October 2009



Tuesday's lecture (20th Oct)

- Introduction to arrays.
- Using arrays for “character-statistics” on text.
- Relationship between arrays and pointers.
- Arrays as parameters to functions.

Today

- Strings.
- Arrays cont. - basic *pattern matching*.
- Bitwise operations on `int` (on board).

Basic data types in C

`int` `char` `float` `double`

Really that's all ...

except for variations such as `signed char`, `unsigned char`, `short`, ...

What about strings?

In computer programming (all languages), a *string* is any *sequence* of characters.

- Many languages offer a string data type.
- C does *not* offer a string data type.
- A string is an *array* of char:
 - *Eg* `char month1[]={'j','a','n','u','a','r','y'};`
 - Or (shorthand) `char month1[]="january";`
 - In a function declaration, as
`return-type function-name(char line[], int length)`

Get *call-by-reference* performance for free for strings.

Pattern matching

We want to write a program that

- ask the user for a pattern
- filters subsequent input for that pattern

Template for reading input

```
int c = getchar();

if (c == EOF) {
    return TRUE;
}
while (c != EOF) {
    /* do something */
    c = getchar();
}
```

Reading input line by line

We want to have a handy function `GetLine` that reads one line from input.

- How do we store the line of text?
- What is the stopping condition of the while loop?
- What happens inside the body?

GetLine()

```
Bool_t GetLine(char line[], int length) {
    int i = 0, c = getchar();
    if (c == EOF) return TRUE;
    while (c != '\n' && c != EOF) {
        if (i < length - 1) {
            line[i] = c;
            ++i;
        }
        c = getchar();
    }
    line[i] = '\0';
    return FALSE;
}
```

”

The big picture

```
char line[LINE_LENGTH], pat[PAT_LENGTH];

GetLine(pat, PAT_LENGTH);
while (!GetLine(line, LINE_LENGTH)) {
    if (IsSubstringOf(pat, line)) {
        PutLine(line);
    }
}
```

Substring matching

- We know how to match characters
- How do we match a substring?

LINE: a test !

PAT: test

test

--> test <-- MATCH!

test

test

Matching condition

First attempt:

```
int j = 0;
while (pat[j] != '\0'
      && pat[j] == text[start+j]) {
    ++j;
}
if (pat[j] == '\0') {
    return TRUE;
}
```

What happens if we run out of characters in text?

Matching condition

Improved:

```
int j = 0;
while (pat[j] != '\0'
      && text[i+j] != '\0'
      && pat[j] == text[start+j]) {
    ++j;
}
if (pat[j] == '\0') {
    return TRUE;
}
```

Do we really need this?

Matching loop

```
Bool_t IsSubstringOf(char pat[], char text[])
/* Returns TRUE iff pat is a substring of text. */
{
    int i = 0, j;

    while (text[i] != '\0') {
        /* match pattern starting at i */
        ++i;
    }
    return FALSE;
}
```