Array Comparison, Strings and Loops

CMPT 125
Mo Chen
SFU Computing Science
17/1/2020
Lecture 5

Today

- Array Comparison
- Strings
- Nested Loops
Array Comparison

- Write a function to compare two arrays
- Array parameters passed by base address
  - Style points: use `int arr[]` instead of `int *arr`

```c
int arrCompare(int A[], int B[], int length) {
    for (int i = 0; i < length; i++) {
        if (A[i] < B[i]) {
            return -1;
        } else if (A[i] > B[i]) {
            return 1;
        }
    }
    return 0;
}
```

array bounds passed separately
Arrays of `char`

- **type `char` is 1 byte per element**
  - traditionally to hold one ASCII character
  - an array of `char` is a string!
- **end of string terminated by null char: `'\0'`**

```c
int main ( ) {
    char msg[10] = "ur n00b!";
    printf("%s\n", msg);
}
```

```
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
msg[10]: 'u' 'r' ' ' 'n' '0' '0' 'b' '!' '\0'
```

```
117 114 32 110 48 48 98 33 0
```
<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>NUL</td>
<td>(null)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>SOH</td>
<td>(start of heading)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>002</td>
<td>STX</td>
<td>(start of text)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>003</td>
<td>ETX</td>
<td>(end of text)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>004</td>
<td>EOT</td>
<td>(end of transmission)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>005</td>
<td>ENQ</td>
<td>(enquiry)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>006</td>
<td>ACK</td>
<td>(acknowledge)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>007</td>
<td>BEL</td>
<td>(bell)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>010</td>
<td>BS</td>
<td>(backspace)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>011</td>
<td>TAB</td>
<td>(horizontal tab)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>012</td>
<td>LF</td>
<td>(NL line feed, new line)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>013</td>
<td>VT</td>
<td>(vertical tab)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>014</td>
<td>FF</td>
<td>(NP form feed, new page)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>015</td>
<td>CR</td>
<td>(carriage return)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>016</td>
<td>SO</td>
<td>(shift out)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>017</td>
<td>SI</td>
<td>(shift in)</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>020</td>
<td>DLE</td>
<td>(data link escape)</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>021</td>
<td>DC1</td>
<td>(device control 1)</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>022</td>
<td>DC2</td>
<td>(device control 2)</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>023</td>
<td>DC3</td>
<td>(device control 3)</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>024</td>
<td>DC4</td>
<td>(device control 4)</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>025</td>
<td>NAK</td>
<td>(negative acknowledge)</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>026</td>
<td>SYN</td>
<td>(synchronous idle)</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>027</td>
<td>ETB</td>
<td>(end of trans. block)</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>030</td>
<td>CAN</td>
<td>(cancel)</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>031</td>
<td>EM</td>
<td>(end of medium)</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>032</td>
<td>SUB</td>
<td>(substitute)</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>033</td>
<td>ESC</td>
<td>(escape)</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>034</td>
<td>FS</td>
<td>(file separator)</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>035</td>
<td>GS</td>
<td>(group separator)</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>036</td>
<td>RS</td>
<td>(record separator)</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>037</td>
<td>US</td>
<td>(unit separator)</td>
<td></td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
int main ( ) {
    int password[3] = {1,2,3};
    int answer[3];

    for (int i = 0; i < 3; i++) {
        printf("Enter digit %d: ", i+1);
        scanf("%d", answer+i);
    }
    if (password != answer) {
        printf("Incorrect password!\n");
    }
}
String Comparison

```c
#include <stdio.h>
#include <string.h>

int main () {
    char password[4] = "abc";
    char answer[4];

    printf("Enter 3-character code: ");
    scanf("%s", &answer);
    if (strcmp(password, answer) != 0) {
        printf("Incorrect password!\n");
    }
}
```

C library function to do string comparisons:
- 0 means equal
- < 0 means first < last
- > 0 means first > last

not &answer because answer is a pointer!
Common String Functions

int strlen(char s[])

- returns the length of the string
- counts characters until null terminator
- Q: What happens if there is no null terminator?

void strcpy(char dest[], char src[])

- copies the string dest[] ← src[]
- Q: What must be true about dest[]?
String I/O

Input

- `scanf("%s", str);`
- `scanf("%[^\n]s", str);`
- `gets(str);`

Output

- `printf("%s", str);`
- `puts(str);`
Nested Loops

- It is possible to include any sequence of statements within a loop body including:
  - calculations
  - function calls
  - if statements
  - other loops
- Just like you did in Python!

Classic Problem: Write a function that scans an array of `int`. It returns 1 if and only if two of the elements are the same, 0 otherwise.
These statements run the most frequently in the worst case
- What is the worst case?
- How many times when length = 4?

```c
int dup_chk(int a[], int length) {
    int i = length;
    while (i > 0) {
        i--;
        int j = i - 1;
        while (j >= 0) {
            if (a[i] == a[j]) {
                return 1;
            }
            j--;
        }
    }
    return 0;
}
```
Another Performance Measure

● Often consider the *worst-case* behaviour as a benchmark
  ○ make guarantees about code performance under all circumstances

● Can predict performance by counting the number of steps required by algorithm in the worst case
  ○ Derive total steps (T) as a function of input size (N)
Analysis

int dup_chk(int a[], int length) {
  int i = length;
  while (i > 0) {
    i--;
    int j = i - 1;
    while (j >= 0) {
      if (a[i] == a[j]) {
        return 1;
      }
      j--;
    }
  }
  return 0;
}

Q. What is N?
- The number of elements in the array

Outside of loop: 2 (steps)

Outer loop: 3N + 1

Inner loop: 3i + 1 for all possible i from 0 to N - 1.
  = 3/2 N^2 - 1/2 N

Grand total = 3/2 N^2 + 5/2 N + 3
A quadratic function!
Empirical Measurement

- Another graph - a quadratic this time!
- Confirms predictions: doubling \((x^2)\) the input size leads to quadrupling \((x^4)\) the running time

<table>
<thead>
<tr>
<th>(N)</th>
<th>time (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>89</td>
</tr>
<tr>
<td>20,000</td>
<td>365</td>
</tr>
<tr>
<td>40,000</td>
<td>1,424</td>
</tr>
<tr>
<td>100,000</td>
<td>9,011</td>
</tr>
</tbody>
</table>
2D Maximum Density Problem

Problem: Given a 2-dimensional array \( (N \times N) \) of integers, find the \( 10 \times 10 \) swatch that yields the largest sum

Applications:
- Resource management and optimization
- Finding brightest areas of photos
Algorithm / Code?

- Simple approach: Try all possible positions for the upper left corner
  - \((N - 9) \times (N - 9)\) of them
  - use a nested loop
- add each swatch using a 10 × 10 nested loop
- A brute-force approach!
  - Generate a possible solution [naively]
  - Test it [naively]
```c
int max10by10(int a[N][N]) {
    int best = 0;
    for (int u_row = 0; u_row < N-10; u_row++) {
        for (int u_col = 0; u_col < N-10; u_col++) {
            int total = 0;
            for (int row = u_row; row < u_row+10; row++) {
                for (int col = u_col; col < u_col+10; col++) {
                    total += a[row][col];
                }
            }
            best = max(best, total);
        }
    }
    return best;
}
```

**Precise accounting:**

\[348N^2 - 6956N + 34762\] operations

**Approximate Method:**

Count the *barometer instructions*, the instructions executed most frequently. Usually, in the innermost loop.

- Innermost loop: \[11 + 10 + 10 = 31\] ops
- Total: \[31 \times 10 \times (N - 9) \times (N - 9) = 310N^2\]