CS31 Discussion

Jie(Jay) Wang

Week6 Nov.4
Outline

• Project 4
• Array
• C Strings
Project 4

- DDL: Monday, November 7
- Read the spec & FAQ carefully
- Incremental development
- You can call your function in other functions
  - *If you wish, your implementation of a function required here may call other functions required here.* (spec)
Project 4

• Return value
  • All functions that return an int must return -1 if they are passed any bad arguments (e.g., a negative array size, or a position that would require looking at the contents of an element past the last element we’re interested in). Unless otherwise noted, passing 0 to the function as the array size is not itself an error; it merely indicates the function should examine no elements of the array.
  • Empty string (i.e., the string “”)
    • Empty string (i.e., the string “”) is just another string, no more special than “hillary” or “donald”.
  • N equals 0
    • Do what makes sense.
  • The empty sequence is a subsequence of every sequence (even an empty one), at position 0.
Array

• Declare an array
  • \texttt{<type> <name>[size]}
    \begin{verbatim}
    int a[4];
    \end{verbatim}

• \texttt{a[i]} is an i-th variable in the array \texttt{a}.

• \texttt{size} should must be a positive integer constant.

\begin{verbatim}
    int a[4];
    const int N = 10;
    int a[N];
\end{verbatim}

• You can treat each element of the array as a variable.

\begin{verbatim}
    x[3] = 5;
    x[1]++;
    cout << x[i] << endl;
\end{verbatim}
Initialization of an Array

```c
int a[5] = {1, 2, 3, 5, 7};
int a[] = {1, 2, 3, 5, 7};
```

- You cannot set the size to what’s less than the number of elements in the initialization statement.
- However, it is okay to set the size to what’s more than the number of elements in the initialization statement.

```c
int a[3] = {1, 2, 3, 5, 7};  // wrong
int a[10] = {1, 2, 3, 5, 7};  // okay
```
Common mistakes

- No `size()` function is defined for arrays.

```c
int a[10];
for (int i = 0; i < a.size(); i++) {
    ...
}
```

```c
const int SIZE = 10;
int a[SIZE];
for (int i = 0; i < SIZE; i++) {
    ...
}
```
Common mistakes

• Out-of-range access not allowed!

```cpp
int a[10];
a[15] = 5; // error
a[-10] = 4; // error
a[9] = 10; // okay
```
Array

• Multi-dimensional arrays

```cpp
#include <iostream>

using namespace std;

int main () {
    int i[ROWS][COLUMNS] = {
        {row_00, row_01, ...},
        {row_10, row11, ...},
        ...,
        {row_i0, row_i1, ...}};

    some2DArray[i][j];
```
Arrays in a Function

```c
void fibo10(int fib[]);
```

Note that the size of fib is not specified, you can explicitly pass the size in the function.

```c
void fibo10(int fib[], int n);
```

Now after you learnt about 2D arrays.

```c
void fibo10(int fib[][], int n);
```
Arrays in a Function

• When C++ functions pass arguments by default for basic types like int, double, and string,
  • Pass by Value
    • The function parameters are copies of the arguments
• Question: Will the code compile? If so, what’s the output?

```cpp
#include <iostream>
#include <string>
using namespace std;

int foo (int a);
int main () {
    int i = 1;
    foo(i);
    cout << i << endl;
}
int foo (int a) {
    return a++;
}
```

Output: 1
Arrays in a Function

• Pass by Reference

```cpp
#include <iostream>
#include <string>

using namespace std;

int foo (int& a);

int main () {
    int i = 1;
    foo(i);
    cout << i << endl;
}

int foo (int& a) {
    return a++;
}
```

Output:

2
Arrays in a Function

• **Question:** Will the code compile? If so, what’s the output?

```cpp
#include <iostream>
#include <string>
using namespace std;

int foo (int a[]);
int main () {
    int i[] = {1, 2};
    foo(i);
    cout << i[0] << endl;
}

int foo (int a[]) {
    return a[0]++;
}
```

Output:
2

Pass by Pointer

```cpp
int i[]  
1 2
```
C Strings

• Why we learn C strings?
• An array of characters
• C strings are **null-terminated**
  • Every C string ends with a marker called the **zero byte** (null character), which we use an escape sequence \0 to represent (its ASCII number is 0).

```c
char s[10];
```
• The maximal length of this string is 9 (not 10)
• Note that for string, there is no null terminator.
C Strings

• To initialize a string

```c
char s[10] = "HOWAREYOU";

| H | O | W | A | R | E | Y | O | U | \0 |
```

```c
char s[] = "HOWAREYOU";

| h | e | l | l | o | \0 |
```

```c
char s[10] = “hello”;

| h | e | l | l | o | \0 |
```

×  ```c
char s[10] = “Hello, world!”;
```

C Strings

• **Question:** Will this code compile? What’s the output?

```c
#include <iostream>
using namespace std;

int main() {
    char s[10] = "HOWAREYOU";
    s[3] = '\0';
    cout << s << endl;
}
```

**Output:**
HOW
C Strings

• **Question:** Will this code compile? What’s the output?

```cpp
#include <iostream>
using namespace std;

int main() {
    char s[10];
    s = "abcdefg";
    cout << s << endl;
}
```

**Output:**

Cannot compile.
Direct assignment like that works only in an initialization expression.
C Strings

• **Question:** Will this code compile? What’s the output?

```cpp
#include <iostream>
#include <string>

using namespace std;

int main () {
    char a[3] = "ca\0";
    cout << a << endl;
}
```

**Output:**
```
c a \0 \0
```
C Strings

• **Question:** Will this code compile? What’s the output?

```cpp
#include <iostream>
#include <string>

using namespace std;

int main () {
    char a[4] = "ca\0";
    cout << a << endl;
}
```

**Output:**

```
c a \0 \0
```
**C Strings**

- **Question**: Will this code compile? What’s the output?

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s = "hello";
    cout << s.size() << ' ';
    s[1] = '\0';
    cout << s.size() << '\n';
}
```

**Output:**
5 5
## Functions for C Strings

- `#include <cstring>`

<table>
<thead>
<tr>
<th>Operation</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>strlen(s)</code></td>
<td>Returns the length of <code>s</code>, not counting ‘\0’.</td>
</tr>
<tr>
<td><code>strcpy(t,s)</code></td>
<td>Copies the string <code>s</code> to <code>t</code>. (Notes: <code>t=s</code> won’t do the job. Also, <code>strcpy</code> doesn’t do the size check for you. You must make sure there’s enough space in <code>t</code> yourself.)</td>
</tr>
<tr>
<td><code>strncpy(t,s,n)</code></td>
<td>Copies the first <code>n</code> characters of <code>s</code> to <code>t</code>. (Note: No size check.)</td>
</tr>
<tr>
<td><code>strcat(t,s)</code></td>
<td>Appends <code>s</code> to the end of <code>t</code>. (Notes: No size check. <code>t += s</code> won’t do the job.)</td>
</tr>
<tr>
<td><code>strcmp(t,s)</code></td>
<td>Compares <code>t</code> and <code>s</code>. Returns 0 if they are equal, something greater than 0 if <code>t &gt; s</code>, and something less than 0 if <code>t &lt; s</code>. (Note: <code>t == s</code> or <code>t &lt; s</code> won’t do the job.)</td>
</tr>
</tbody>
</table>
C Strings

• Two alternatives to traverse a C string.

```c
char s[10] = "HOWAREYOU";
for (int k = 0; t[k] != '\0'; k++)
    cout << t[k] << endl;
```

```c
#include<cstring>
...

char s[10] = "HOWAREYOU"
for (int k = 0; k < strlen(s); k++)
    cout << t[k] << endl;
```
C Strings

• An array of C strings

```c
char s[10][20];
```

• In `s`, we can store up to **10** C strings, and each C string can be at most **19** characters long.

```c
s[3];
// refer to the string in position 3
```

```c
s[3][5];
// refer to the letter in position 5 of the string in position 3
```
C Strings

• Convert a C string into a C++ string, and vice versa.

```cpp
char cs[10] = "hello";
string cpps;
cpps = cs;

string cpps = cs;
string ccpps(cs);
```

C string to string

```cpp
char cs[10];
string cpps = "hello";
strcpy(cs, cpps.c_str());
```

string to c string

• You cannot:

```cpp
char cs[10] = cpps.c_str();
```

```cpp
char cs[10];
cs = cpps.c_str();
```
C Strings

• **Question:** Will this code compile? What’s the output?

```c
#include <iostream>
#include <cstring>

using namespace std;
int main () {
    char a[] = "sup";
    char b[] = "hey\0you";
    char c[] = {"\0"};
    cout << strlen(a) << endl;
    cout << strlen(b) << endl;
    cout << strlen(c) << endl;
}
```

**Output:**
3
3
0
#include <iostream>
#include <cstring>

using namespace std;

int main () {
    char a[] = "sup";
    char A[] = "SUP";
    cout << strcmp(a, A) << endl;
    cout << strcmp(A, a) << endl;
    return 0;
}

Output:
32 -32
C Strings

• **Question**: Will this code compile? What’s the output?

```cpp
#include <iostream>
#include <cstring>
using namespace std;
int main () {
    char a[100] = "sup\0";
    char b[] = " my bro?";
    strcat(a, b);
    cout << a << endl;
}
```

**Output**: sup my bro?
Programming practice

• Design a function, csReverse, that takes in a cstring and reverses the order of characters in it.

  void csReverse (char c[]);
Programming practice

• Design a function, multiCStringAppend, that takes in an array of cstrings and a single cstring, and appends toAppend to each of n elements of target.

```c
void multiCStringAppend (  
    char target[][MAX_CSTRING_SIZE],  
    char toAppend[],  
    int n);
```
Credit to 2 previous CS31 TAs

• This slide is finished with reference from:
  • Andrew Forney
  • http://web.cs.ucla.edu/~forns/
  • Brian Choi
  • http://netlab.cs.ucla.edu/~schoi/cs31/
Thanks!