CS 31 Discussion 1A, Week 6

Zengwen Yuan (zyuan [at] cs.ucla.edu)
Humanities A65, Friday 10:00—11:50 a.m.
Today’s focus

• Array

• C-string

• Style choice
Intro to arrays

- An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.
- In C++, indexing starts from 0.
- Syntax: `<type> <name>[size]`;
- Array sizes must be defined at compile time and must be constant.
#include <iostream>
#include <string>
using namespace std;

int main() {

    // An array of called yarn that can hold 2 strings
    // uninitialized
    string yarn[2];

    // An array called foo that can hold 3 ints
    // initialized with values 0, 0, 0
    int foo [3] = {};  

    // An array called bar that can hold 3 ints
    // initialized with values 1, 2, 3
    int bar [] = {1, 2, 3};
}
#include <string>
using namespace std;

int main() {
    // declaration of a 2D array
    int myArray [3][5];

    // assign values (initialization)
    int foo [4][2] = {{0, 1},
                      {3, 2},
                      {3, 5},
                      {3, 8}};

    // equivalently this: foo is equal to bar
    int bar [4][2] = {0, 1, 3, 2, 3, 5, 3, 8};
}
Array: assign value and access elements

• An element in an array is accessed by using the subscripts:

```c
int val_1 = my1dArray[3];
int val_2 = my2dArray[0][1];
```

• An element in an array is assigned value similarly:

```c
// 1D array initialization
for (int i = 0; i < SIZE; i++) {
    my1dArray[i] = 0;
}

// 2D array initialization
for (int j = 0; j < ROW_SIZE; j++) {
    for (int k = 0; k < COL_SIZE; k++) {
        my2dArray[j][k] = 0;
    }
}
```
Array: representation in memory
Arrays and functions

• Recall:
  • call-by-value
  • call-by-reference

• What about array as an argument?

```c
double getAverage(int arr[], int size);
```

• What if I want to return an array from a function?
  • Currently you do not know how to do it, yet… ଘ(œ)器官
Pitfall: constants

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

int main () {
    const int arr[3] = {1, 2, 3};

    // Is it ok?
    arr[1] = 5;
    cout << arr[1] << endl;
}
```
C-string: introduction

• Motivation:
  • C doesn't have the std::string type
  • convenience v.s. performance

• Method: use arrays of char

• Zero-byte termination: the null character is always represented as '\0' and has character code 0.
C-string: declaration and initialization

```c
char s[10];  // 10 unitialized chars
char t[10] = { 'h', 'e', 'l', 'l', 'o', '\0' };  // tedious
char t[10] = "hello";  // means exactly the same as above
```
#include <iostream>
#include <string>
using namespace std;

int main () {
    const int MAX_CSTR_SIZE = 6;
    char taker[MAX_CSTR_SIZE] = "<:)~";
    char rot[] = ">:)"

    // The mysteriously vanishing devious face
    for (int i = 0; taker[i] != '\0'; i++) {
        int j = i;
        do {
            cout << rot[j++];
        } while (j < 5 && rot[j] != '\0' && rot[j] == taker[j]);
        cout << endl;
    }
}

courtesy: Andrew Forney
C-string: quiz

• Write a loop to assign \( t \) to \( s \)

• Write a loop to compare two strings \( t \) and \( s \)
C-string: library functions


- `strlen` returns the number of characters before the first null character in a cstring. It is similar to string.length() method.

- `strcmp` returns a number that determines if the first argument is less than (some value < 0), greater than (some value > 0), or equal to (= 0) the second.

- `strcpy` takes the cstring in the source and copies it into the destination.

- `strcat` adds the second argument to the end of the first argument, similar to the + operator defined on C++ strings.
C-string: quiz

• 1. implement `strlen()`

• 2. what is the output? Why?

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

int main () {
    char lower[] = "sup";
    char upper[] = "SUP";

    cout << strcmp(lower, upper) << " 
    cout << strcmp(upper, lower);
}
```

A. 10 -10  
B. -10 10  
C. 32 -32  
D. -32 32
Style choices

• Styles matter

• Stick to one / follow guidelines of your organization.
  
  • https://google.github.io/styleguide/cppguide.html
  
  • http://geosoft.no/development/cppstyle.html
Style: a good example

```cpp
bool isValidUppercaseStateCode(string stateCode)
{
    const string codes =
        "AL.AK.AZ.AR.CA.CO.CT.DE.DC.FL.GA.HI.ID.IL.IN.IA.KS."
        "KY.LA.ME.MD.MA.MI.MN.MS.MO.MT.NE.NV.NH.NJ.NM.NY.NC."
        "ND.OH.OK.OR.PA.RI.SC.SD.TN.TX.UT.VT.VA.WA.WV.WI.WY";
    return (stateCode.size() == 2 &&
            stateCode.find('.') == string::npos && // no '.' in stateCode
codes.find(stateCode) != string::npos); // match found
```
Style: a not so good example...

```java
while (isdigit(pollData[i])) {
    voteDigit += pollData[i];
    i++;
}

if (voteDigit.size() == 1) {
    vote = (pollData[i-1] - '0'); // conversion from char to int
}
else if (voteDigit.size() == 2) {
    vote = (pollData[i-2] - '0')*10 + (pollData[i-1] - '0'); // conversion from char to int
}
}
if (vote == 0) {
    return 2;
}
```