CS 31 Discussion 4

Tianxiang (Peter) Li

tianxiang@cs.ucla.edu

Using comments

- Blocks of code to perform certain task

```
  // Get and validate initial meter reading

cout << "Initial meter reading: ";
int initial;
cin >> initial;
cin.ignore(10000, '\n');
if (initial < 0)
{
    cout << "---" << endl
         << "The initial meter reading must be nonnegative." << endl;
    return 1;
}

  // Get and validate final meter reading

cout << "Final meter reading: ";
int final;
cin >> final;
cin.ignore(10000, '\n');
```

```
// Get and validate initial meter reading

// Get and validate final meter reading

// Get and validate customer name

// Get and validate month

// Determine applicable thresholds and rates for the season

// Compute bill

// Print bill
```

Using comments

- Logic of computation

```
    // Determine applicable thresholds and rates for the season

    double tier1Threshold;
    double tier2Rate;

    if (month >= 4  &&  month <= 10)  // April through October, high season
    {
        tier1Threshold = HIGH_TIER1_THRESHOLD;
        tier2Rate = HIGH_TIER2_RATE;
    }
    else // low season
    {
        tier1Threshold = LOW_TIER1_THRESHOLD;
        tier2Rate = LOW_TIER2_RATE;
    }
```

Some notes on project report

- Report
  - How to write test cases, example:

Negative initial meter reading (-1, 35, Peter, 12)
Final meter less than initial meter (5, 4, Peter, 8)
Empty string for customer name (0, 1, , 8)
Month number not between 1 and 12 (0, 1, Peter, 13)

Invalid Input

Low season first tier (0, 5, Peter, 1)
Low season second tier (0, 50, Peter, 1)
High season first tier (0, 5, Peter, 6)
High season second tier (0, 50, Peter, 6)

Valid Input, test different conditional statements

Some notes on homework

Problem 2 marking criteria

- (The problem said to write a brief, simple English sentence)
  - 2 marks off for description longer than 30 words
  - 1 mark off for inaccurate description
- Example answer: *It prints a diagonal line of the specified length.*

Some notes on homework

Problem 4 marking criteria

- Check len somewhere in your code
  - When len <=0 no output
- Use do-while for outler loop
- Use while for inner loop
- Syntax errors

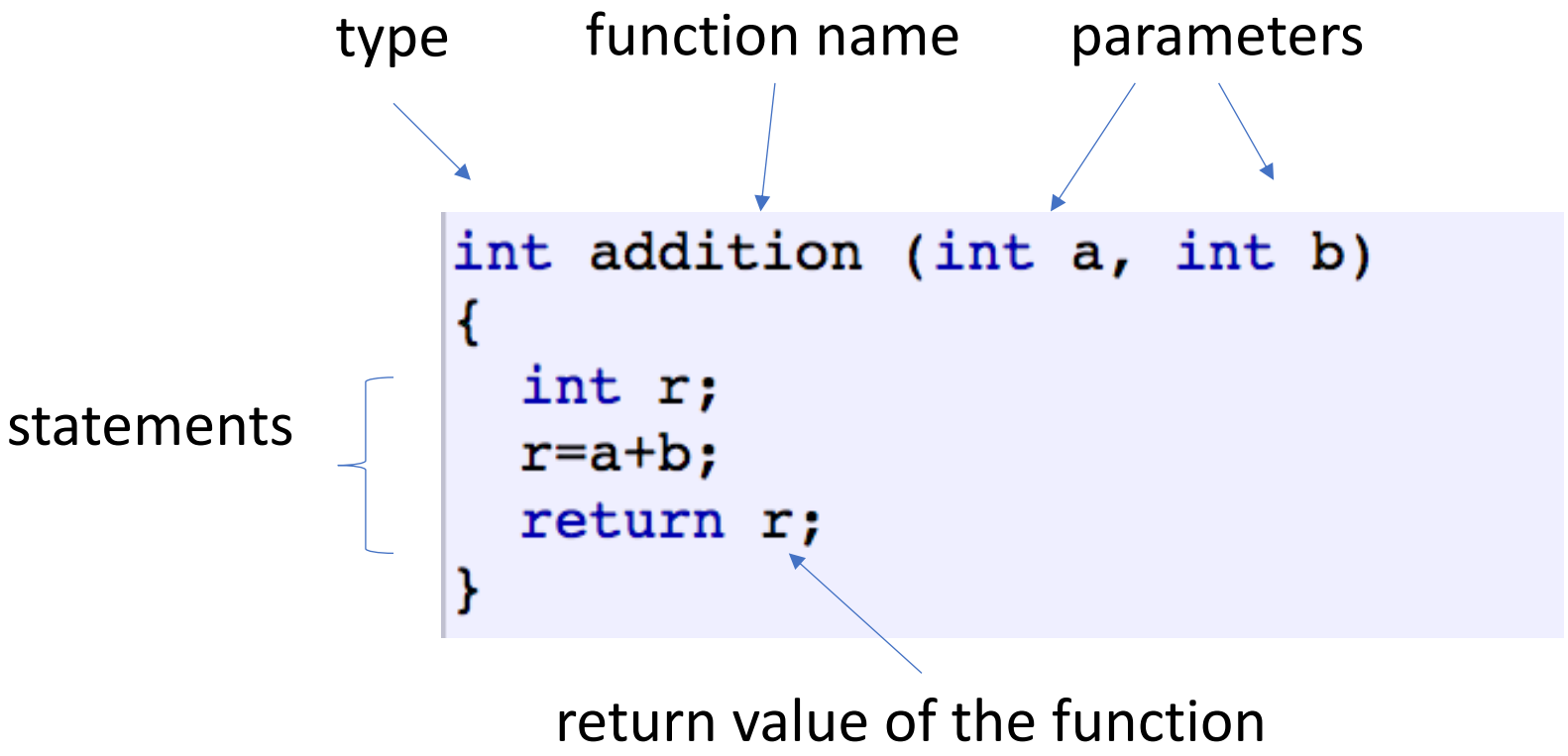
```cpp
#include <iostream>
using namespace std;

int main()
{
    int len;

    cout << "Enter a number: ";
    cin >> len;

    if (len > 0)
    {
        int i = 0;
    do
        {
        int j = i+1;
        while (j < len)
        {
            cout << ' ';
            j++;
        }
        cout << "#" << endl;
        i++;
    } while (i < len);
    }
}
```

functions


type
function name
parameters
int addition (int a, int b)
{
int r;
r=a+b;
return r;
}
statements
return value of the function

defining functions

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

using namespace std;

int func1(int x, string str)
{
    // function definition omitted
}

int main()
{
    int y = func1(5, “hello”);
    ...
}
```

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

using namespace std;

int func1(int x, string str);

int main()
{
    int y = func1(5, “hello”);
    ...
}

int func1(int x, string str)
{
    // function definition omitted
}
```

Calling a function

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

// function declaration
int max(int num1, int num2);

int main () {
   // local variable declaration:
   int a = 100;
   int b = 200;
   int ret;

   // calling a function to get max value.
   ret = max(a, b);
   cout << "Max value is : " << ret << endl;

   return 0;
}

// function returning the max between two numbers
int max(int num1, int num2) {
   // local variable declaration
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

declare function

call function

define function

Exercise

4) Write a function *integerDivide* that does integer division without using the division operator (/). You can assume both parameters will be positive.

For example:

integerDivide(6, 2) returns 3 integerDivide(7, 2) returns 3

Call-by-value vs call-by-reference

```
1   #include <iostream>
2   #include <string>
3   using namespace std;
4
5   void change(int data);
6
7   int main()
8   {
9     int data = 3;
10    change(data);
11    cout << "Value of the data is: " << data<< endl;
12    return 0;
13  }
14
15  void change(int data)
16  {
17    data = 5;
18  }
```

Call-by-value vs call-by-reference

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

void change(int &data);

int main()
{
  int data = 3;
  change(data);
  cout << "Value of the data is: " << data<< endl;
  return 0;
}

void change(int &data)
{
  data = 5;
}
```

Call by reference

parameters

- call-by-value parameters
  - Only the **copy of the value** of the argument is plugged in

```
void showResults(int output1, int output2);
```

- call-by-reference parameters
  - A reference to the argument is plugged in, used to **access the actual argument**
  - Variable's value can be changed by the function

```
void getInput(double &receiver);
```

which is equivalent to

```
void getInput(double& receiver);
```

Call by reference in detail

- Program variables are implemented as memory locations
- When compiled, first_num is stored in 1008, with value 0

```
int first_num = 0, second_num = 0;
```

Memory Location	Value	
...		
1008		
1010	0	← first_num
1012	0	← second_num
1014		
...		

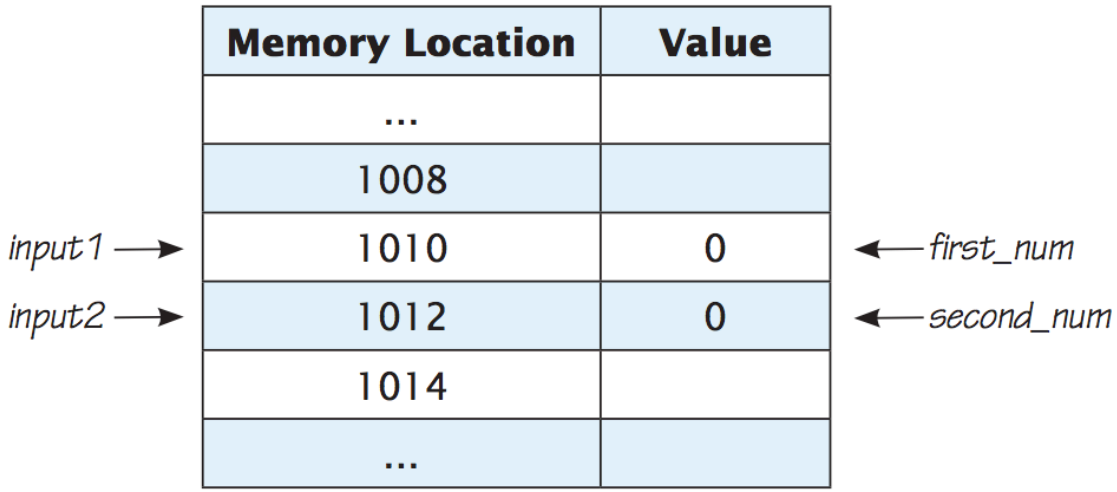
Call by reference in detail

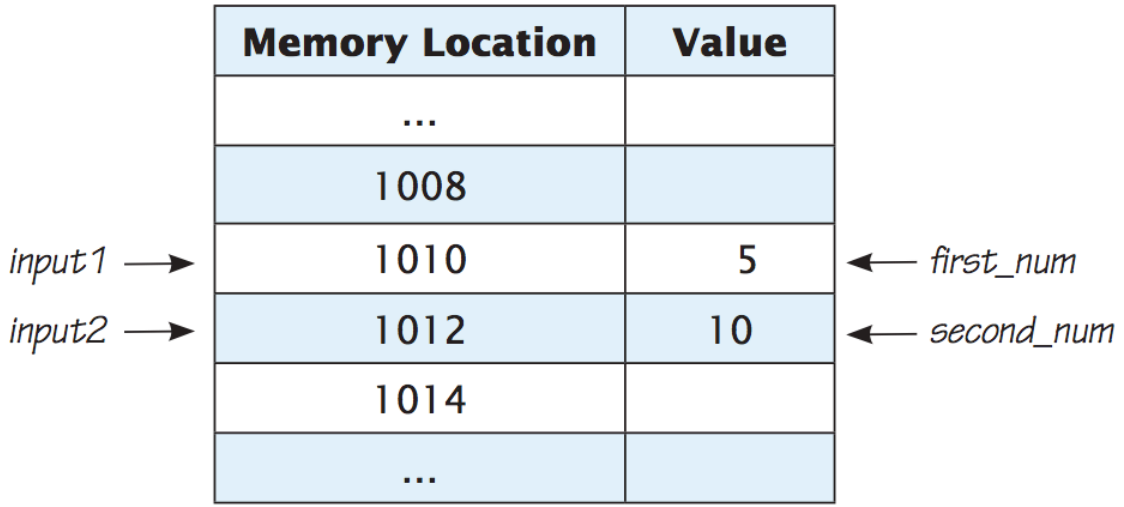
- When function is executed
  - It gives the **memory locations** associated with first_num, second_num
  - its is **NOT** returning the **copy of the values**

```
void get_numbers(int& input1, int& input2);
```

```
int main( )
{
    int first_num = 0, second_num = 0;

    get_numbers(first_num, second_num);
```





Passing By Reference to Const

- const keyword marks the reference to be non-modificable
- Value accessed by reference
- Const reference similar to passing argument by value
- Increased efficiency for parameter of large types, avoid making copies

```
string concatenate (const string& a, const string& b)
```

operation	what it does	example
`string s = "hello";` `string s2 = "!!!";`	declares strings `s` and `s2`	
`s.length() or s.size()`	returns the length of `s`	`cout << s.size(); // prints 5`

`s[i] or s.at(i)`	returns `i`-th character
	`i` must be an integer between 0 and size-1 (inclusive).
		`cout << s[1];  // prints 'e'`
		`cout << s.at(0);  // prints 'h'`

`s.empty()`	returns true if `s` is empty	`if (!s.empty())` `  cout << "not empty";`

`s + s2`
`s + "?"`
	concatenates two strings	`cout << s + s2;  // prints "hello!!!"`
		`cout << s + "?";  // prints "hello?"`

`s.substr(i, n)`
`s.substr(i)`
	takes a substring of length n, starting from the `i`-th character	`cout << s.substr(2,2);  // prints "ll"`
		`cout << s.substr(2);  // prints "llo"`

`s.replace(i, n, s2)`	replaces a substring of length `n` starting at `i` with another string s2, and sets `s` with a new string	`s.replace(2, 2, s2); // sets s to`
		`// "he!!!o"`

There are a few more functions, but this set of functions should be enough for our purposes.

Now here are some functions you can call on characters, after including `<cctype>` library:

operation	what it does
`char c;`	declares a character `c`
`isspace(c)`	true if `c` is a whitespace character (space, tab, newline)
`isalpha(c)`	true if `c` is a letter
`isalnum(c)`	true if `c` is a letter or digit
`islower(c)`	true if `c` is a lowercase letter
`isupper(c)`	true if `c` is an uppercase letter
`tolower(c)`	returns the lowercase version of `c`, if `c` is a letter
`toupper(c)`	returns the uppercase version of `c`, if `c` is a letter

Example code

11. What is the output of the following program?

```
string justLetters(string s)
{
    string result = "";
    for (int k = 0; k != s.size(); k++)
    {
        if ( islower(s[k]) )
            result += s[k];
        if ( isupper(s[k]) )
            result += tolower(s[k]);
    }
    return result;
}

int main()
{
    cout << justLetters("CS 31 began September 28.") << endl;
}
```

Exercise

5) Write a function findLastLength that takes in a string that consists of uppercase alphabetical characters, lowercase alphabetical characters, and empty space ' ' characters. It returns the length of the last word, unless the last word does not exist, in which case it returns 0.

For example:
findLastLength("Misfits should have won against SKT") returns 3
findLastLength(" ") returns 0

String comparison

- String comparison is done in lexicographical order
  - i.e. the order they appear in the dictionary
- Lowercase letter ALWAYS greater than upper case letters
- Alphabets are always greater than digits

```
"Zebra" < "alphabet"
"zebra" > "alphabet"
"uniform" < "university"
"alpha" < "alphabet"
"10apples" < "7apples"
```

```
string s1;
string s2;
cout << "Enter two strings: ";
getline(cin, s1);
getline(cin, s2);
if (s1 < s2)
   cout << s1 << " is smaller than " << s2 << endl;
else if (s1 == s2)
   cout << s1 << " is equal to " << s2 << endl;
else
   cout << s1 << " is greater than " << s2 << endl;
```

Exercise

- What is the output?

(**Note:** `str.substr(str.size())` returns an empty string.)

```
void mystery(int pos, string &str)
{
  if (pos < 0 || pos >= str.size())
    return;
  cout << str[pos] << " ";
  str = str.substr(0, pos) + str.substr(pos + 1);
}

int main()
{
  string s = "abcdefg";
  mystery(3, s);
  mystery(3, s);
  mystery(4, s);
  mystery(1, s);
  cout << endl;
  cout << "s = " << s << endl;
}
```

This program prints:

```
d e g b
s = acf
```

Exercise

- What is the output?

This program prints:

```
d e g b
s = acf
```

(**Note:** `str.substr(str.size())` returns an empty string.)

```
void mystery(int pos, string &str)
{
  if (pos < 0 || pos >= str.size())
    return;
  cout << str[pos] << " ";
  str = str.substr(0, pos) + str.substr(pos + 1);
}

int main()
{
  string s = "abcdefg";
  mystery(3, s);
  mystery(3, s);
  mystery(4, s);
  mystery(1, s);
  cout << endl;
  cout << "s = " << s << endl;
}
```

This program prints:

```
d e g b
s = acf
```

Common mistakes

When calling a function, you only need the values, not the types.

```
int main()
{
    int y;
    y = func1(int 5, string “hello”);     // WRONG!
    y = func1(x = 5, str = “hello”);      // WRONG!
    y = func1(5, “hello”);                // CORRECT

    return 0;
}
```

Common mistakes

Do not define a function within a function, since you can't!

```
int func1(int x, string str)
{
   void func2(int x, string str)
   {
      ...
   }
   ...
}
```

Project 3

- Goal
  - verify that a proposed course works
- Grid
  - rectangular grid of up to 25 x 25
  - each cell is either empty or a wall
- Car Movement
  - if a car is at an empty cell, it can move only to an empty adjacent cell (north, east, south, and west)
- Course segment
  - One letter (N,E,S,W, n,e,s,w)
  - Zero, one, or two digit characters

  Example: N2e1Es2e1

```
   1234
1  ...*
2  .*..
3  .*..
```

Example 3X4 grid

Walls:
(1,4),(2,2),(3,2)

Project 3

bool isCourseWellFormed(string course)

- Check whether course is syntactically valid
  - return True/False
- String iteration
  - Check index
- If Syntax correct
  - Print out sentence

Project 3

int driveSegment(int r, int c, char dir, int maxSteps)

- Return number of steps [0, maxStep] the car can travel in direction *dir* starting at position (*r* , *c*)
  - Travel up to maxSteps, stop if hit wall or reach border
- Return -1 for invalid arguments

Project 3

int driveCourse(int sr, int sc, int er, int ec, string course, int& nsteps)

- Check input arguments
  - If input argument invalid
    - returns 2 and leaves nsteps unchanged
- Check if car can reach (er,rc) starting from (sr,sc) following course
  - if car able to finish the complete course
    - set nsteps to the number of steps taken
    - If car end up at (er,ec)
      - Return 0
    - else
      - Return 1
  - else (If car hit wall or go off grid before finish course)
    - Return 3 and set nsteps to max number of steps the car can travel

Project 3

```
int driveCourse(int sr, int sc, int er, int ec, string course, int& nsteps)
{
        //check arguments
        if ( isCourseWellFormed(course) )
        …

        /*
            check each step of course, see if a car can move in the specified
            direction dir for the specified number of steps maxSteps
            starting from the point (r, c)
        */
        int opt = driveSegment( r, c, dir, maxSteps)
        //check opt, can the car finish the course?
        //also need to set current position (r,c) after each step
        //at the end check whether (r,c) is (er,ec) when all steps finish
}
```

Exercise – String Manipulation

8) Write a function that takes a string and (1) reverses the lowercase vowels, then (2) converts all lowercase vowels into uppercase.

Example: reverseLowercaseVowels("begin projects early") should return "bAgEn prEjOcts IErly" reverseLowercaseVowels("hellO world") should return "hOllO wErld" [Notice that the capital O in the original string did not change positions because you only need to reverse all the lowercase vowels. That is, keep the uppercase vowels in the same position.]

Q & A