Lecture 5
Fall 2018
Prof. Zhang

Last class:

1. Variables: type, size, address

address of a variable: the location of the variable in the memory, & operator

you can envision memory to be a large array of bytes, and address to be the index of the first byte used to store a variable

scope: the part of the code/program where a variable name is accessible (visible)
depending on where the variable is defined
local variable:
global variable:
block variable:

lifetime: (during the program's execution) the time from a variable is created to the time when the variable is deleted (memory returned to system)
static lifetime: global variable, static variable
automatic lifetime: local variable and block variable
dynamic lifetime: variables that are created using new/malloc, deleted using delete/dealloc

Q: How can you access another function's local variable inside a function?

```
void swap (int & x, int & y)
        // two int parameters passed-by-reference (&)
{
    int tmp=x;
    x = y;
    y = tmp;
}

int main()
{
    int a=10,b=20;
    swap (a,b);  //we pass a,b by reference to swap, allowing swap() function to access
        //a and b, local variables of main
}
```

2. Array Review

* Partially filled array: length <= capacity

* Keep track how many elements are stored in array
e.g., int Numers[5]={2,3,4};
int NumbersLen=3;

or using a special value, **terminator** (that won't appear as a valid value for array element) to indicate end of array

e.g., int Numbers[5]={2,3,4,-1};
char greetings[6]={'h','e','l','l','o','\0'};
// '\0' is same as 0, a special char to terminate a char array
same as char greetings[6]="hello";

Today

1. Passing arrays into functions:

* array variable stores the starting address of the array
* passing an array parameter to a function is passing the address of the array (i.e., the called function will be accessing the actual argument).

In a sense, we can think array parameters are always passed-by-reference (i.e., the function accesses the actual argument provided by the caller, not a copy).

2. Lab3 assignment

* Why stores the digits in the "reversed order"?

// 1234 will be stored as an array of int
int num1[4]={4, 3, 2, 1};
int num1_len=4;
// 56 will be stored as
int num2[100]={6,5};
int num2_len=2;

* In order to add more easily (align 1-th digits, 10-th digits … when adding two numbers)

* How to add?

-> To simplify the logic, you can pad the number with 0's on the left (pad array with 0's in the end)

-> Implement addition algorithm that you have mastered since grade school

* How to read a large integer?

* Read into a char array (C string) or C++ string class

```
string input;
cout <<"Enter large int (up to 100 digits):";
cin >> input;
cout <<"You enter:"<<input<<endl;
```

* Process string from right to left… (one's digit to higher digit):

```
len = input.length();
//todo: make sure  len <=100 …
for (int i=0;i<len;i++)
{
        num[i]=input[len-i-1]-'0';   //last digit first
                //convert char '0', '1', '2' … '9' to number 0, 1, 2, …9
                //take advantage of ASCII code arrangement

        if (num[i]>9 || num[i]<0)
        {
                cout <<"invalid char " << input[len-i-1]<<endl;
                return false;
        }
}
```

```
//lab exercise 1 solution
#include <iostream>
using namespace std;

/* display int array terminated using -1
 precondition: the array has been set up appropriately,
   i.e., -1 is stored in the array, at an index in the proper range
 postcondition: ...
 */
void DisplayIntArrays (int b[ ])
//                    the type of parameter b is "array of int"
{
   cout << "address of b[0]:" << &(b[0])<<endl;

   for (int i=0;b[i]!=-1;i++)
      cout <<b[i]<<" ";

   cout <<endl;
}

/* (partially) filled an array with int numbers
 parameters: the array, and its capacity (size)
 return the actual number of int stored in the array
*/
int ReadIntegers (int c[ ], int CAP)
{
     for (int i=0;i<CAP-1;i++)
    {
         cin >> c[i];
         if (c[i]==-1)
              break;
    }
    c[i]=-1;
    return i;

}
/* reverse array of int that is terminated using -1
*/
void ReverseIntArray (int d[])
{
   // find the index of last element in array d
   int last=0;
   while (d[last]!=-1)
      last++;

   int front=0;
   int end=last-1; //last points to the last element of array

   while (front<end)
   {
      int tmp = d[front];
      d[front] = d[end];
      d[end] = tmp;
      front++;
      end--;
   }

}
```

```
int main()
{
   int a[200]={19,23,37,-1};

   cout <<"address of a[0]:"<< &(a[0]) << endl;
   cout <<“a:”<<a<<endl;
   DisplayIntArrays (a);
   //            a is an array of int

   ReadIntegers (a, 200);

   //….

}
```

$./a.out
address of a[0]:0x7ffee6fa3570
a: 0x7ffee6fa3570
address of of b[0]:0x7ffee6fa3570
19 23 37