University of Hertfordshire

7COM1025
Programming for Software Engineers
Lecture 8

Dr. Renato C. de Amorim

THE ONE HUNDRED UNDER FIFTY

BASIC: GLOBAL VARIABLES

```
#include<iostream>
using namespace std;
void func1();
void func2();
int count;
int main()
{
    for (int i=0; i<10;i++)
    {
        count = i*2;
        func1();
    }
    return 0;
}
```

```
void func1()
{
    cout<<"count: "<<count<<endl;
    func2();
}
void func2()
{
    int count;
    for (count=0; count<3; count++)
        cout<<'.';
}
```

POINTERS AS PARAMETERS

```
#include<iostream>
using namespace std;
void f(int *j);
int main()
{
    int i, *p;
    p=&i;
    f(p);
    cout<<i<<endl;
    return 0;
}
void f(int *j)
{
    *j=100;
}
```

```
#include<iostream>
using namespace std;
void f(int *j);
int main()
{
    int i;
    f(&i);
    cout<<i<<endl;
    return 0;
}
void f(int *j)
{
    *j=100;
}
```

ARRAYS AS PARAMETERS

When an array is an argument to a function, the address of the first element of the array is passed, not a copy of the entire array.
There are three solutions:

```
#include<iostream>
using namespace std;
void display(int num[10]);
int main()
{
    int t[10], i;
    for (i=0;i<10;++i) t[i]=i;
    display(t);
    return 0;
}
void display(int num[10])
{
    for(int i=0; i<10; i++)
        cout<<num[i]<<' ';
}
```

```
#include<iostream>
using namespace std;
void display(int num[]);
int main()
{
    int t[10], i;
    for (i=0;i<10;++i) t[i]=i;
    display(t);
    return 0;
}
void display(int num[])
{
    for(int i=0; i<10; i++)
        cout<<num[i]<<' ';
}
```

```
#include<iostream>
using namespace std;
void display(int *num);
int main()
{
    int t[10], i;
    for (i=0;i<10;++i) t[i]=i;
    display(t);
    return 0;
}
void display(int *num)
{
    for(int i=0; i<10; i++)
        cout<<num[i]<<' ';
}
```

PROBLEM 8.1

Re-write your encryption program so that you have two functions, one to encrypt and one to decrypt for each of the three ways to pass a C style string.

STRING AS PARAMETER

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
void invertCase(char *str);
int main() {
    char str[80];
    gets(str);
    invertCase(str);
    cout<<str;
}
void invertCase(char *str) {
    while(*str)
    {
        if(isupper(*str)) *str=tolower(*str);
        else if(islower(*str)) *str=toupper(*str);
        str++;
    }
}
```

RETURNING POINTERS

```
#include <iostream>
using namespace std;
char *get_substr(char *sub, char *str);
int main(){
    char *substr;
    substr = get_substr("two", "one two three");
    cout<<"Substring found: "<<substr;
    return 0;
}
```

```
char *get_substr(char *sub, char *str){
    char *p, *p2, *start;
    for (int t=0; str[t]; t++)
    {
        p=&str[t];//reset pointers
        start=p;
        p2=sub;
        while (*p2 && *p2==*p){
            p++;
            p2++;
            }
        if(!*p2)
            return start;//beginning of substring
    }
    return 0; //no match found
}
```

MAIN() ARGUMENTS

argc is an int holding the number of arguments (min=1).
argv is a pointer to an array of character pointers. Each pointer in argv points to a string containing a command-line argument.

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[]){
    for (int i=0; i<argc;i++)
            cout<<argv[i]<<endl;
    return 0;
}
```

RECURSION

A function can call itself!

```
#include <iostream>
int my_pow(int base, unsigned int exponent);
using namespace std;
int main(){
    int a;
    unsigned int b;
    cout<<"Enter base and exponent: "<<endl;
    cin>>a;
    cin>>b;
    cout<<"Result: "<<my_pow(a, b)<<endl;
    return 0;
}
```

```
int my_pow(int base, unsigned int exponent)
{
    if (exponent == 0)
        return 1;
    else
        return base * my_pow(base, exponent - 1);
}
```

PROBLEM 8.2

Remember the factoring problem?
Re-write it so that it uses functions and recursion.

VECTORS

```
#include <iostream>
#include<vector>
using namespace std;
void showall(vector<int> vec);
int main()
{
    int num, index;
    vector<int> my_vector;
    do {
            cout<<"Enter a number: "<<endl;
            cin>>num;
            if (num!=-1)
                my_vector.push_back(num);
    }while (num!=-1);
    cout<<"You entered "<<my_vector.size()<< " numbers."<<endl;
    cout<<"Enter an index: "<<endl;
    cin>>index;
    cout<<"At index "<<index<<": "<<my_vector[index]<<endl;
    cout<<"You vector values: ";
    showall(my_vector);
    cout<<"Index of the item you want to remove: "<<endl;
    cin>>index;
    my_vector.erase(my_vector.begin()+index);
    cout<<"You vector values: ";
    showall(my_vector);
    cout<<"Now you have "<<my_vector.size()<<"
numbers."<<endl;
    return 0;
}
```

```
void showall(vector<int> vec)
{
    for(vector<int>::iterator item =vec.begin(); item!=vec.end();item++)
        cout<<*item<<' ';
    cout<<endl;
}
```

VECTOR OF VECTORS

```
#include <iostream>
#include<vector>
using namespace std;
void showall(vector<vector<int> > vec);
int main()
{
   int row, col, num,index;
   vector<vector<int> > my_vector;
   cout<<"Enter the number of rows and columns: "<<endl;
   cin>>row>>col;
   my_vector.resize(row);
   for(int i = 0; i<row;i++)
      for (int ii=0; ii<col;ii++)
      {
         cout<<"Enter element ("<<i+1<<","<<ii+1<<"): ";
         cin>>num;
         my_vector[i].push_back(num);
      }
   showall(my_vector);
   cout<<"Enter row to remove: "<<endl;
   cin>>index;
   my_vector.erase(my_vector.begin()+index);
   showall(my_vector);
   return 0;
}
```

```
void showall(vector<vector<int> > vec)
{
   for(register int i=0; i<vec.size(); i++)
   {
      for(register int ii=0; ii<vec[i].size(); ii++)
         cout<<vec[i][ii]<<' ';
      cout<<endl;
   }
}
```