Chapter 3

Organization

3.1 Functions

3.1.1 Functions included in standard library

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

int main () {
    const double PI=acos (-1.0);
    double x,y;
    x=2.0;
    y=exp (x*PI);
    cout<<y<<endl;
    return 0;
}
```

3.1.2 User defined functions

In general this is

```
Functions

data_type function_name(data_type variable1, data_type variable 2...)
{
    code
    return (constant or variable with the same data_type as function)
}
```

How to Compile Multiple Files

```
g++ FUNCTION EXAMPLE.cpp FUNCTIONS.cpp –o FUNCTION EXAMPLE
OR

g++ –c FUNCTION EXAMPLE.cpp

g++ –c FUNCTIONS.cpp

g++ FUNCTIONS.o FUNCTION EXAMPLE.o –o FUNCTION EXAMPLE
```

PARAMETERPASSING EXAMPLE.cpp

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

double square1 (double a)
{
    a = pow(a, 2.0);
    return a;
}

double square2 (double& a)
{
    a = pow(a, 2.0);
    return a;
}

int main () {
    double x, y;
    x = 2.0;
    y = square1(x);
    cout << x << " , " << y << endl;
    x = 2.0;
    y = square2(x);
    cout << x << " , " << y << endl;
    return 0;
}

3.2 Multiple Files

PARAMETERPASSING_EXAMPLE.cpp

#include <iostream>
#include <cmath>
#include "squares.h"
using namespace std;

int main () {
    double x, y;
    x = 2.0;
    y = square1(x);
    cout << x << " , " << y << endl;
    x = 2.0;
    y = square2(x);
    cout << x << " , " << y << endl;
    return 0;
}

squares.cpp

#include <cmath>
```
double square1 (double a)
{
    a=pow(a,2.0);
    return (a);
}
double square2 (double& a)
{
    a=pow(a,2.0);
    return (a);
}

3.3 Preprocessor

#define PI 3.14 //replaces PI with 3.14 wherever it appears
const int PI =3.14; //normally safer
#define square (x) ((x)*(x)) //not anywhere there is an x you should use (x),
    //if x=2+3 you want ((2+3)*(2+3)) not (2+3*2+3)

#ifdef DEBUG
    cout<<"DEBUG MODE, "<<"output some debug info like parameter values"
#endif /* DEBUG */
    //If DEBUG is not defined it is just like everything in between is commented out

#define DEBUG /* turns on debugging */
#undef DEBUG /* turns off debugging */

#ifndef is not defined

#include "file.h" //for user made files

#ifndef _FILE_H_INCLUDED_
#define _FILE_H_INCLUDED_
#include "file.h"
#endif /* _FILE_H_INCLUDED_ */
    //This way you don't include the same thing multiple times if multiple
    //files all want to include or define something. This is crucial if the file
defines data structures.