Week 6

EMT 101 – Engineering Programming

Dr. Farzad Ismail

School of Aerospace Engineering
Universiti Sains Malaysia
Nibong Tebal 14300 Pulau Pinang

Functions and Subroutines

- Writing a code with only a main function is efficient or less efficient?
- Efficiency can be measured more than one way.
- Efficiency in terms of program management and hence debugging
- Can also be measured in terms of how much data is being passed from one part of the program to another.

Functions versus subroutines

- It will be messy if you include all your algorithm in the main
- Use subroutines or functions
- This add additional cost (and perhaps errors) due to information passing of variables and identifiers
- Need to declare functions and subroutines in header files

Example

- Solving a fluid flow over an airfoil
- In one main function
- In a main function with many subfunctions or subroutines
- Pseudo-code presentation

1 Main function

```
int main()
  {  initializations
     …..
     geometry and grid modelings
      ….
     mathematical and physical models
      ……
     loop to solve the problem for pressure and velocities
     ……..
     plot the velocity and pressure fields
     ……..
     return 0;
   }
```

Problems with 1 Main functions

- Main program becomes messy
- Difficult to debug
- The program is very ‘rigid’ since any changes to part of the program requires going through the whole main program
- Difficult to extend the code to the purposes

How to overcome?

- Use ‘divide and conquer’ approach in managing the main programs into smaller subprograms
- Use either functions or subroutines

Main function + subroutines

```
Global declarations
int main()
  {   subroutines for initializations

      subroutines for geometry and grid modelings

      subroutines for mathematical and physical models

      subroutines  to solve the problem for pressure &velocities

      subroutines for plot the velocity and pressure fields

      return 0;
  }
```

Discussions on how the program and subprograms work

- The algorithm for each subroutine lies outside main
- Requires the declaration of the subroutines either in the same main file or another file which can be read and compiled together with the main file (similar to using a C++ library)
- Example subroutine for geometric model:

  geometric_model (input parameters, grid size, coordinates, boundary conditions…..)
- Each subroutine requires information as input to perform

The advantages with using subroutines

- The whole program is more manageable in terms of program developments and debugging
- The program is now flexible to changes, i.e. for example if another geometry is used, no need to change main function, just change the subroutine function
- The subroutines can be used with other main functions (compatibility with other code)
- However, there is a disadvantage of using subroutines…

Kinetic Energy Program Using Function

```
#include <iostream>
#include <string>
#include <cmath>
#include <conio.h>
using namespace std;

double Compute_KE(double u, double v)
{
double KE=0.0;
KE= u*u + v*v;
return KE;
}

int main()
{
   int n;
   cout << "Enter number of particles:  " << endl;
   cin >>n;
 double KE[n]; double u[n]; double v[n];
  for (int i=0; i<n; i++)
      {
       cout << "Enter the u velocity for particles: " << endl;
       cin >> u[i];
       cout << "Enter the v velocity for particles: " << endl;
       cin >> v[i];

       KE[i]= Compute_KE(u[i],v[i]);
       cout << "KE of particle " << i+1 << " is " << KE[i] << endl;
       }
       getch();
return 0;
}  // end of program body
```

Kinetic Energy Program Using Subroutine

```
#include <iostream>
#include <string>
#include <cmath>
#include <conio.h>
using namespace std;

//Declaration of a subroutine to be used in Main
void Determine_KE(int n, double u[], double v[], double KE[], double TKE)
 {
    for (int i=0; i<n; i++)
      {
       KE[i]= u[i]*u[i] + v[i]*v[i];
       TKE += KE[i];
      }
 cout  << "TKE is " << TKE << endl;
 }

int main()
{   int n;
   double TKE;
   cout << "Enter number of particles:  " << endl;
   cin >>n;
 double KE[n]; double u[n]; double v[n];
  for (int i=0; i<n; i++)
      {
       cout << "Enter the u velocity for particles: " << endl;
       cin >>u[i];
       cout << "Enter the v velocity for particles: " << endl;
       cin >> v[i];
      }
 Determine_KE(n,u,v,KE, TKE);
return 0;
}
```

Tutorial 1

- Using subroutines write a program to perform a numerical integration of

  f(x)=x^2*sin(x)*exp(x^2) over x=[0,Pi] having a choice of

  (i) rectangular rule

  (ii) the Simpson's rule (take home)

  Divide the domain into N subsections, where N=5,10,20,40. Compare your results.

Assignment

- Please refer to Homework 2 in website