PH5720: Numerical Methods and Programming

Week - 02

How to program

A few words on C++ and STL

Prabhat R. Pujahari

Indian Institute of Technology Madras

A few words on C++ and STL

Relational Operator

Name — Symbol

- Equal ==
- Not equal !=
- Greater than >
- Greater than or equal >=
- Less than <
- Less than or equal <=

Logical Operator

Operator — Symbol

- AND &&
- OR ||
- NOT !

A few words on C++ and STL

❖ Increment Operator

- *One wants to increase the variable by 1 unit*

```
xx = xx +1;

xx += 1;

xx++;        // postfix increment op

++xx;        // prefix increment op
```

❖ Decrement Operator

- *One wants to decrease the variable by 1 unit*

```
xx = xx -1;

xx -= 1;

xx--;        // postfix decrement op

--xx;        // prefix decrement op
```

A few words on C++ and STL

❖ Program # 01 : sum1.cpp

❖ Requirement

$$result = \sum_{i=1}^{100} i$$

❖ Loop - Syntax

```
int sum = 0
for( i = 0; i < 101; i++)
   {
       sum = sum + i;
    }
```

❖ How to do it?

```
int sum = 1;
sum = sum + 1;
sum = sum + 2;
.
.
.
.
sum = sum + 100;
```

A few words on C++ and STL

- Program # 01 : sum1.cpp

```
#include <iostream>
int main()
{
     using std::cout;
     using std::endl;

     int i;
     int result = 0;

     for (int i = 0, i < 101; i++)
        {
           result +=  i;
        }

     cout << " result = " << result << endl;
```

- Requirement

$$result = \sum_{i=1}^{100} i$$

- Loop - Syntax

```
for( i = 0; i < 100; i++)
   {

    statement;
   }
```

A few words on C++ and STL

❖ Program # 02 : sum2.cpp

```
#include <iostream>
int main()
{
      using std::cout;
      using std::endl;

      int i;
      int result = 0;

      while ( i < 101)
         {
            result +=  i;
            i++;
         }

      cout << " result = " << result << endl;


      return 0;
}
```

❖ Requirement

$$result = \sum_{i=1}^{100} i$$

❖ Loop - Syntax

```
while( condition )
   {

     statement;
   }
```

A few words on C++ and STL

❖ Program # 03 : sum3.cpp

```
#include <iostream>
int main()
{
      using std::cout;
      using std::endl;

      int i;
      int result = 0;

      do
         {
            result +=  i;
            i++;
         }
      while (i < 101);

      cout << “ result = “ << result << endl;


      return 0;
}
```

❖ Requirement

$$result = \sum_{i=1}^{100} i$$

❖ Loop - Syntax

```
do
   {

      statement;
   }
While ( condition );
```

A few words on C++ and STL

❖ Loop - Syntax

```
for( i = 0; i < 100; i++)
   {

     statement;
   }
```

1st

```
while( condition )
  {

    statement;
   }
```

2nd

```
do
   {

     statement;
   }
While ( condition );
```

3rd

- If the condition is true then the loop continues in 1 and 2
- The statements are executed first and then checks the condition in 3

❖ Program # 04 : decision1.cpp

```
#include <iostream>
int main()
{
    using std::cout;
    using std::endl;

    int result = 1;

    for (int i = 1; i < 11; i++)
      {
        if (decision == 0)
          { result += i; }
        if (decision == 1)
          { result *= i; }
      }

    cout << " result = " << result << endl;


    return 0;
}
```

❖ Decision Loop - Syntax

```
if( decision == 0)
  {

    statement;
  }
```

❖ Requirement

$$result = \sum_{i=1}^{10} i$$

$$result = \prod_{i=1}^{10} i$$

❖ Program # 05 : decision2.cpp

```
#include <iostream>
int main()
{
      using std::cout;
      using std::endl;

      int result = 0;

      for (int i = 1, i < 11; i++)
          {
             if (decision == 0)
               { result += i; }
             else if (decision == 1)
               { result *= i; }
          }

      cout << " result = " << result << endl;


      return 0;
}
```

❖ Decision Loop - Syntax

```
if( decision == 0)
    {
      statement;
     }
else if (decision)
    {
      statement;
    }
```

❖ Requirement

$$result = \sum_{i=1}^{10} i$$

$$result = \prod_{i=1}^{10} i$$

❖ Program # 06 : decision3.cpp

```
#include <iostream>
int main()
{
    using std::cout;
    using std::endl;

    int decision = 0;

    if (decision == 0)
      { cout << " Today is Monday" << endl; }
    else if (decision == 1)
      { cout << " Today is Tuesday" << endl; }
    else if (decision = 2)
      { cout << " Today is Wednesday" << endl; }
    else if (decision == 3 )
      { cout << " Today is Thursday" << endl; }

    cout << " result = " << result << endl;

    return 0;
}
```

❖ Nested Decision Loop - Syntax

```
if( decision )
  {
    statement;
  }
else if ( decision )
  {
    statement;
  }
else if ( decision )
  {
    statement;
  }
```

❖ Program # 07 : switch.cpp

```
#include <iostream>
int main()
{
    using std::cout;
    using std::endl;

    int decision = 0;

    switch (decision)
     {
      case 0:
         cout << " Today is Monday" << endl;
         break;
      case 1:
         cout << " Today is Tuesday" << endl;
         break;
      default:
         cout << " Not defined, Try again " << endl;
     }
    return 0;
}
```

❖ "switch" statement

```
switch( decision )
  {
    case 0:
       statement;
       break;
    case 1:
       statement;
       break;
    case 2:
       statement;
       break
    case 3:
       statement;
       break;
    default:
    statement;
  }
```

❖ Program # 07 : switch.cpp

```
#include <iostream>
int main()
{
      using std::cout;
      using std::endl;

      int decision = 0;

      switch (decision)
        {
          case 0:
              cout << " Today is Monday" << endl;
              break;
          case 1:
              cout << " Today is Tuesday" << endl;
              break;
          default:
              cout << " Not defined, Try again " << endl;
       }
      return 0;
}
```

❖ How does it work?

- *Depending on the "switch" statement, It switches to the right one. But if you do not "break" it, it goes sequentially.*

- *It is not necessary to use "default". But some of the compilers complain. Then it is better to use it.*

❖ Program # 08 : compare.cpp

```
#include <iostream>
int main()
{
    using std::cout;
    using std::endl;

    int alpha = 3;
    int beta   = 5;
    int min;

    if (alpha < beta)
        min = alpha;
    else
        min = beta;

    cout << " Minimum = " << min << endl;

    return 0;
}
```

❖ Conditional operator

- The decision operator is used when one compares with two values and decides one value

For example: Out of 3 and 5, which is smaller?

❖ Conditional operator

```
#include <iostream>
int main()
{
	using std::cout;
	using std::endl;

	int alpha = 3;
	int beta   = 5;
	int min;

	if (alpha < beta)
		min = alpha;
	else
		min = beta;

	cout << " Minimum = " << min << endl;

	return 0;
}
```

- The decision operator is used when one compares with two values and decides one value

  For example: Out of 3 and 5, which is smaller?

- In C++, there exists a conditional op.

```
min = (alpha < beta) ? alpha : beta;
```

- If the parenthesis is true, then alpha is minimum or beta is minimum.

```
#include <iostream>
int main()
{
      using std::cout;
      using std::endl;

      int alpha = 3;
      int beta   = 5;
      int min;

      min = (alpha < beta) ? alpha:beta;

      cout << " Minimum = " << min << endl;

      return 0;
}
```

❖ Conditional operator

- The decision operator is used when one compares with two values and decides one value

  For example: Out of 3 and 5, which is smaller?

- In C++, there exists a conditional op.

- If the parenthesis is true, then alpha is minimum or beta is minimum.

❖ Program # 09 : infinite.cpp

```
#include <iostream>
int main()
{
      using std::cout;
      using std::endl;

      int i, result = 0;

      for (;;)
         {
            result +=  i;
            if (I == 101)
             {
              cout << " result = " << result << endl;
              exit(0);
             }
            i++
         }
      return 0;
}
```

❖ Infinite Loop - Syntax

```
for(;;)
   {
    statement;
   }
```

❖ Requirement

When one does not know

$$result = \sum_{i=1}^{100} i$$

Inside the parentheses, any integer Value one can give. Even negative Integer also.
It does not take floating point value.

A few words on C++ and STL

❖ Program

```
#include <iostream>
using namespace std;
int main()
{

    float length, width;

    [ ................
      block of code to calculate
      area of a rectangle
    ]


    return 0;
}
```

❖ Functions

```
float AreaRectangle(float len, float wid)
{
    statement;
    return area;
}
```

❖ Program 10 : arearect1.cpp

❖ Functions

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

float AreaRectangle(float len, float wid)
{
   float area = len*wid;
   return area;
}

int main()
{
      float length = 0, width = 0;
      float area = 0;
      length = 10.; width = 5.;

      area = AreaRectangle(length, width);

      cout << “area = “ << area << endl;

      return 0;
}
```

```
float AreaRectangle(float len, float wid)
{
   float area = len*wid;
   return area;
}
```

The same program can be done in a different way.

```
#include <iostream>
using namespace std;
float AreaRectangle(float len, float wid);

int main()
{
      float length = 0, width = 0;
      float area = 0;
      length = 10.; width = 5.;

      area = AreaRectangle(length, width);

      cout << “area = “ << area << endl;
      return 0;
}

float AreaRectangle(float len, float wid)
{
   float area = len*wid;
   return area;
}
```

```
float AreaRectangle(float len, float wid)
{
   float area = len*wid;
   return area;
}
```

❖Program #11: arearect3.cpp

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

float AreaRectangle( );
float AreaRectangle(float len, float wid);

int main()
{
      float length = 0., width = 0.;
      float area = 0.;
      length = 10.; width = 5.;

      area = AreaRectangle();
      cout << “area = “ << area << endl;

      area = AreaRectangle(length, width);

      cout << “area = “ << area << endl;

      return 0;
}
```

❖ Overloaded Functions

```
float AreaRectangle( )
{
   float length = 20.0;
   float width   = 10.0;
   float area = length * width;
   return area;
}

float AreaRectangle(float len, float wid)
{
   float area = len*wid;
   return area;
}
```

‘Polymorphism’ feature in C++

❖ Program #12: arearect4.cpp

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

inline float AreaRectangle(float len, float wid);

int main()
{
      float length = 0, width = 0;
      float area = 0;
      length = 10.; width = 5.;

      area = AreaRectangle(length, width);

      cout << “area = “ << area << endl;

      return 0;
}
```

❖ Inline Function - Syntax

```
inline float AreaRectangle(float len, float wid)
{
   return len*wid;
}
```

❖ Recommendation

If there is few lines (1 or 2 lines) codes in the function

❖ Advice

If you do not know, then go for the normal function definition.

❖ Analysis

❖ Function

```
float AreaRectangle(float len, float wid)
{
    return len*wid;
}
```

- For normal function, the compiler creates a set of instruction. If you are calling a function, it jumps to that place and executes it and comes back to the original place.
- That means only one set of copy exists.
- Some overhead because of jumping back and forth.

❖ Inline Function - Syntax

```
inline float AreaRectangle(float len, float wid)
{
   return len*wid;
}
```

- If you call “inline function” many times, then it inserts that many times.
- That means it keeps that many number of copies.
- The executable becomes big. It will take more time.

❖ Program #13: array.cpp

❖ Container - Syntax

int aa[5]; // stores integer value

float xx[5]; // stores real value

❖ Example

To store real or integer value for further processing

```
#include <iostream>
int main()
{
    using std::cout;
    using std::endl;
    int i = 0;
    int aa[5];
    float xx[5];
    for (i=0; i< 5; i++)
      {
        aa[i] =  i;
        xx[i] = (float) i + 0.5;
      }
    for (i=0; i< 5; i++)
      {
        cout << " aa[" << i << "]=" << aa[i]
             << " xx[" << i << "]=" << xx[i] << endl;
      }

    return 0;
}
```

❖ Arrays: Arrays are known as container

❖ Syntax: int a[10];

❖ It can be integer, float, character etc.

❖ How to initialize it?

```
a[0] = 1;
a[1] = 2;
.
.
a[9] = 10;
```

or

```
a[10] = {1, 2, ….., 10};
```

or

```
for (int i = 0; i < 10; i++)
{
    a[i] = i;
}
```

❖ Multidimensional array: int a[5][3];

❖ Initialization:

```
a[5][3] = { {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9},
            {10, 11, 12}
            {13, 14, 15} };
```

or

```
int ii = 0;
for (int i = 0; i < 5; i++)
{
   for (int j = 0; j < 3; j++)
   {
    a[i][j] = ii;
    ii++;
}
```

❖ Example: Passing Multidimensional Array to the function

❖ Program 14 : array2.cpp

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

void Print(int aa[ ][3])
{
   for (int i = 0; i < 5; i++)
      {
         for (int j = 0; j < 3; j++)
             cout << aa[i][j] << endl;
      }
}

int main()
{
      int a[5][3];
      Print(a);

      return 0;
}
```

Systems of Linear Equations

Reference: Numerical Recipes in C++, W.H. Press, et. al.

- Systems of linear equations arise in many contexts. There solution is a fundamental tool in numerical methods.

To be discussed in this chapter

- Several Matrix related topics.
- The solution of linear equations.
- Computing determinant of a matrix.
- Calculate the matrix inverse of a square matrix.

- Consider a system of M linear equations in N unknowns

$$a_{11}x_1 + a_{12}x_2 + \ldots. \ a_{1N}x_N = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots. \ a_{2N}x_N = b_2$$

$$a_{31}x_1 + a_{32}x_2 + \ldots. \ a_{3N}x_N = b_3$$

$$.$$

$$.$$

$$a_{M1}x_1 + a_{M2}x_2 + \ldots. \ a_{MN}x_N = b_M$$

a_{ij}, b_i are known constants

x_j is unknown

Matrix notation

- In vector notation: **A . x = b**

- For a square matrix (N equations and N unknowns), the behavior is critically affected by the determinant,

$|A| = \sum \varepsilon_{i_1 i_2 i_3 .. i_N} a_{1i_1} a_{2i_2} a_{3i_3} \ldots a_{Ni_N}$

where the sum is computed with $i_1, i_2, i_3 \ldots i_N$ each taking all possible values over the range 1 to N, and

$\varepsilon =$ -1 if $i_1, i_2, i_3 \ldots . i_N$ is an odd permutation of 1,2,3,…N (e.g. as in ε_{2134})

$\varepsilon =$ +1 if $i_1, i_2, i_3 \ldots . i_N$ is an even permutation of 1,2,3,…N (e.g. as in ε_{2143})

$\varepsilon =$ 0 otherwise (i.e. any two indices are the same)

There are N^N terms in the sum, of which N! are non-zero

Singular versus non-singular matrices

- If M = N, there is a unique solution for **x,** namely $\mathbf{x} = \mathbf{A}^{-1} . \mathbf{b}$, if and only if $|\mathbf{A}| \neq 0$.

  Matrices with $|\mathbf{A}| \neq 0$ are said to be non-singular:
  Matrices with $|\mathbf{A}| = 0$ are singular (and they have no inverse)

- Example: consider a diagonal square matrix,

$$\mathbf{A} = \begin{pmatrix} a_{11} & & & & \\ & a_{22} & & & \\ & & a_{33} & & \\ & & & \ldots & \\ & & & & a_{NN} \end{pmatrix}$$

Singular versus non-singular matrices

$$\mathbf{A}^{-1} = \begin{pmatrix} 1/a_{11} & & & & \\ & 1/a_{22} & & & \\ & & 1/a_{33} & & \\ & & & \dots & \\ & & & & 1/a_{NN} \end{pmatrix}$$

exists iff all the a_{ii} are non-zero, so that a unique solution $\mathbf{x} = \mathbf{A}^{-1}.\mathbf{b}$ exists

But if one or more a_{ii} are zero, $|\mathbf{A}| = \Pi a_{ii} = 0$, the matrix is singular, and $\mathbf{A}^{-1}$ does not exist

Matrix properties

Relations	Name	matrix elements
$\mathbf{A} = \mathbf{A}^T$	symmetric	$a_{ij} = a_{ji}$
$\mathbf{A} = \left(\mathbf{A}^T\right)^{-1}$	real orthogonal	$\sum_k a_{ik} a_{jk} = \sum_k a_{ki} a_{kj} = \delta_{ij}$
$\mathbf{A} = \mathbf{A}^*$	real matrix	$a_{ij} = a_{ij}^*$
$\mathbf{A} = \mathbf{A}^\dagger$	hermitian	$a_{ij} = a_{ji}^*$
$\mathbf{A} = \left(\mathbf{A}^\dagger\right)^{-1}$	unitary	$\sum_k a_{ik} a_{jk}^* = \sum_k a_{ki}^* a_{kj} = \delta_{ij}$

Summary:

1. Diagonal if $a_{ij} = 0$ for $i \neq j$,

2. Upper triangular if $a_{ij} = 0$ for $i > j$, which for a 4×4 matrix is of the form

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{nn} \end{pmatrix}$$

3. Lower triangular if $a_{ij} = 0$ for $i < j$

$$\begin{pmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

4. Upper Hessenberg if $a_{ij} = 0$ for $i > j + 1$, which is similar to a upper triangular except that it has non-zero elements for the first subdiagonal row

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{pmatrix}$$

5. Lower Hessenberg if $a_{ij} = 0$ for $i < j+1$

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

6. Tridiagonal if $a_{ij} = 0$ for $|i-j| > 1$

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{pmatrix}$$

For a real $n \times n$ matrix $\mathbf{A}$ the following properties are all equivalent

1. If the inverse of $\mathbf{A}$ exists, $\mathbf{A}$ is nonsingular.
2. The equation $\mathbf{Ax} = 0$ implies $\mathbf{x} = 0$.
3. The rows of $\mathbf{A}$ form a basis of $\mathbb{R}^n$.
4. The columns of $\mathbf{A}$ form a basis of $\mathbb{R}^n$.
5. $\mathbf{A}$ is a product of elementary matrices.
6. 0 is not an eigenvalue of $\mathbf{A}$.

The basic matrix operations that we will deal with are addition and subtraction

$$\mathbf{A} = \mathbf{B} \pm \mathbf{C} \Longrightarrow a_{ij} = b_{ij} \pm c_{ij},$$

scalar-matrix multiplication

$$\mathbf{A} = \gamma \mathbf{B} \Longrightarrow a_{ij} = \gamma b_{ij},$$

vector-matrix multiplication

$$\mathbf{y} = \mathbf{A}\mathbf{x} \Longrightarrow y_i = \sum_{j=1}^{n} a_{ij} x_j,$$

matrix-matrix multiplication

$$\mathbf{A} = \mathbf{B}\mathbf{C} \Longrightarrow a_{ij} = \sum_{k=1}^{n} b_{ik} c_{kj},$$

transposition

$$\mathbf{A} = \mathbf{B}^T \Longrightarrow a_{ij} = b_{ji},$$