MA517M-Basic Programming Laboratory

Laboratory 4 : Repetition and Arrays

Panchatcharam Mariappan[1]

[1]Associate Professor
Department of Mathematics and Statistics
IIT Tirupati, Tirupati

August 28, 2025

Increment/Decrement Operators

Increment/Decrement Operators

Increment (++) and decrement (- -) operators change the value of the operand (integer variable) by 1. Assume $x = 10$;

Operator	Description	Example	Result
`++`	Increases the integer value by 1	`x++`	11
`- -`	Decreases the integer value by 1	`x- -`	9

Bitwise Operators

- These operators perform bit-level operations.
- For example, If $a = 10$ and $b = 15$, then their binary format is $a = 00001010$ and $b = 00001111$.
- Now, &, |, ^ , ~denote respectively AND, OR, XOR, and Complement operators.
- For example, c=a&b = 0000 1010.

Bitwise Operators

Operator	Description	Example	Result
&	Binary AND operator	c = a & b	0000 1010
|	Binary OR operator	c = a|b	0000 1111
^	Binary XOR operator	c = a ^ b	0000 0101
~	Binary complement (Unary)	c = ~a	1111 0101
<<	Binary Left shift operator	c = a<<2	0010 1000
>>	Binary Right shift operator	c = a>>2	0000 0010

`for` **Loop**

`for` **Loop Syntax**

```
for ( initialization; condition; updateStatment )
{
   statement(s);
}
```

`for` **Loop**

- Repetition structure
- Initialization: declare and initialize any loop variables. Executed once
- Condition: Evaluated at each step. If the condition is true, the body of the loop is executed. Otherwise, it jumps out of the loop
- UpdateStatement: Evaluated at each step. Once the body of the loop is executed, the loop variable is incremented or decremented, or updated
- Entry controlled loop

`while` **Loop**

`while` **Loop Syntax**

```
while (condition)
{
  statement(s);
}
```

`while` **Loop**

- Repetition structure
- Condition: Evaluated at each step. If the condition is true, the body of the loop is executed. Otherwise, it exits from the loop
- Entry controlled loop

`do..while` **Loop**

`do..while` **Loop Syntax**

```
do
{
   statement(s);
}while (condition);
```

`for` **Loop**

- Similar to a while loop, but exit exit-controlled loop. That is, the body of the loop is executed at least once, irrespective of whether the condition is true or false.

Examples

Examples

Write a C++ program to calculate the sum of the first n natural numbers using a for loop.

```
#include<iostream>
using namespace std;
int main()
{
  int n,sum;
  cout<<"Enter the integer to get the sum of 1 to n: "<<endl;
  cin>>n;
  if (n<=0)
  {
    cout<<"Invalid Input"<<endl;
    return 0;
  }
  sum=0;
  for(int i=0;i<n;i++)
  {
    sum+=i;
  }
  cout<<"The sum first "<<n<<" natural numbers  =  "<<sum<<endl;
  return 0;
}
```

Examples

Write a C++ program to calculate the sum of the first n natural numbers using a while loop.

```cpp
#include<iostream>
using namespace std;
int main()
{
  int n, sum;
  cout<<"Enter the integer to get the sum of 1 to n: "<<endl;
  cin>>n;
  if (n<=0)
  {
    cout<<"Invalid Input"<<endl;
    return 0;
  }
  sum=0;
  int x=n;
  while (n!=0)
  {
    sum+=n;
    n--;
  }
  cout<<"The sum first "<<x<<" natural numbers  =  "<<sum<<endl;
  return 0;
}
```

Examples

Write a C++ program to get the number of voters (at least 6). Each voter should vote for one of the following choices.

1. Cup
2. Candle
3. Chalk
4. Pen
5. Pencil

Count the number of votes for each choice and then print.

Examples

Count the number of votes for each choice and then print.

```
#include <iostream>
using namespace std;
int main()
{
  int n, vote;
  int cup=0,candle=0,pen=0,pencil=0,chalk=0;
  cout<<"Enter the number of voters "<<endl;
  cin>>n;

  int i=0;
  while(i<n)
  {
    cout<<"Voter "<<i+1<<" : Enter one of the following numbers to Vote: "<<endl;
    cout<<"1. Cup"<<endl;
    cout<<"2. Candle"<<endl;
    cout<<"3. Chalk"<<endl;
    cout<<"4. Pen"<<endl;
    cout<<"5. Pencil"<<endl;
    cin>>vote;
```

Examples

Count the number of votes for each choice and then print.

```
    switch(vote)    {
      case 1:
        cup++; i++;
        break;
      case 2:
        candle++;i++;
        break;
      case 3:
        chalk++;i++;
        break;
      case 4:
        pen++;i++;
        break;
      case 5:
        pencil++;i++;
        break;
      default:
      cout<<"Invalid Input, ReEnter"<<endl;
    } }
  cout<<"Number of Votes for Cup: "<<cup<<endl;
  cout<<"Number of Votes for Candle: "<<candle<<endl;
  cout<<"Number of Votes for Chalk: "<<chalk<<endl;
  cout<<"Number of Votes for Pen: "<<pen<<endl;
  cout<<"Number of Votes for Pencil: "<<pencil<<endl;
  return 0;}
```

Examples

Write a C++ program to find all prime numbers between 1 to n.

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

  cout<<"Enter an integer to find prime numbers between 1 to n: "<<endl;;
  cin>>n
  cout<<"Prime numbers between 1 to "<<n<< "are\n:";
  for(int i=2; i<=n; i++){
    Prime = 1;
    for(int j=2; j<=i/2; j++){
      if(i%j==0){
        Prime = 0;
        break; }}
    if(Prime==1)
          cout<<i<<"\t";
  }
  cout<<endl;
return 0;
}
```

Comparison

Comparison

- How do you modify the above program using a do..while loop?
- Create a table to list the differences between the for loop, the while loop, and the do..while loop.
- What are definite conditions and indefinite conditions?
  In general, there's no strict rule that one loop type is inherently better than the other. It's about choosing the loop type that fits the specific problem you're trying to solve. If you know the exact number of iterations, a for loop might be more appropriate. If the number of iterations depends on a condition, a while loop might be the better choice.

Comparison

For loops:

1. **Definite iteration:** for loops are particularly useful when you know the exact number of times you want to iterate through a block of code. They are great for iterating over collections like lists, arrays, or ranges.
2. **Simple counting:** When you want to iterate a fixed number of times, such as performing an action on each element of a list.
3. **Sequential iteration:** When you need to process elements in a sequential manner, like going through the characters in a string.

Comparison

While loops:

1. **Indefinite iteration:** while loops are used when you don't know in advance how many times the loop needs to execute. They continue running as long as a specified condition is met.
2. **Dynamic conditions:** When the loop termination depends on some dynamic condition that may change during the loop's execution.
3. **User input validation:** For repeatedly prompting a user for input until they provide a valid response.

Comparison

- Too many nesting levels will be difficult to understand
- Better to avoid controlling counting loops with floating-point variables
- Control counting loops with integer values
- Place only expressions involving the control variables in the initialization and increment sections of a for statement
- Using commas instead of semicolons in a for header is a syntax error
- Infinite loops: When the condition in the for or while loop never becomes false. To prevent infinite loops, ensure that you do not place a semicolon Immediately after a while statement's header. Make sure the The control variable is incremented (or decremented) in the loop.
- Better to avoid a change of variable in the body of the for loop
- Don't confuse with == operator and = operator in conditions

Arrays

Arrays

- Sequential collection of homogeneous data
- Contiguous memory locations
- One Dimensional
- Multidimensional
- Access elements by indices
- Possible to initialize during declaration
- It is identified by square brackets

Arrays

To declare an array, you need four things:

1. Variable Name
2. data type
3. size of the array or length of the array
4. Dimension of the array (single dimensional or multi-dimensional)

Arrays

Example for arrays: vectors, matrices, tensors

- In Mathematics and physics, you know scalars, vectors (do not confuse with C++ vectors, it meant for vectors of mathematics), matrices and tensors.
- Declaring a variable like `int x = 5;` are scalar type
- Vectors can be represented using one-dimensional array

Syntax

```
1   cType arrayName [arraySize];
```

Arrays

- Declaring a variable like `int x[100] = {0};` is similar to defining a vector with 100 dimensions
- Here, 100 integer variables are declared and initialized to zero at one go.
- All these 100 variables are to be allotted in a contiguous way.
- To refer to a particular location or element in the array, specify the array's name and the position number of the particular element in the array.
- In C++, the indexing starts from 0
- * The first element is referred by `VariableName[0]` Example: `x[0]`
- The i th element is referred by `VariableName[i]` Example: `x[i]`

Examples

Write a C++ program to add two vectors (Vectoraddition.cpp)

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
  float x[50],y[50],res[50];
  for(int i=0;i<50;i++)
  {
    x[i]=0.0+i*0.5;
    y[i]=0.0+i*0.5;
  }
  for(int i=0;i<50;i++)
  {
    res[i]=x[i]+y[i];
  }
  for(int i=0;i<50;i++)
  {
    cout<<x[i]<<" + "<<y[i]<<" = "<<res[i]<<endl;
  }
  return 0;
}
```

Examples

Write a C++ program to conduct an election poll of 11 candidates with 50 voters. Announce the result of the winning candidate.

```
#include<iostream>
using namespace std;
int main(void)
{
  int candidate[11]={0};
  int voters[50]={1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
  1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
  5, 6, 7, 5, 6, 4, 8, 6, 8, 10,7,5,6,7,8,9,7,2,1};
  int winner=0;
  for(int i=0;i<50;i++)
  {
    ++candidate[voters[i]];
  }
  cout<<"Candidates\t\t Votes\n";
  for(int i=0;i<11;i++)
  {
    cout<<i+1<<"\t\t"<<candidate[i]<<endl;
  }
//Write a code to compute the winner by finding maximum number of votes registered candidate
  cout<<"The winner is Candidate"<< winner+1<<"and votes are "<<candidate[winner]<<endl;
  return 0;
}
```

2D Array Syntax

```
1 datatype VariableName[RowSize][ColSize];
```

2D Arrays

1. Declaring a variable like `int x[100][100] = {0}` are similar to defining a vector with 100 dimensions
2. Here, 10000 integer variables are declared and initialized to zero at one go.
3. All these 10000 variables will be allocated in a contiguous way.
4. To refer to a particular (i,j)th element of the array, use `x[i][j]`
5. In C++, the indexing starts from 0
6. The first element is referred by `VariableName[0][0]` Example: `x[0][0]`

Examples

Write a C++ program to add two matrices (Matrixaddition.cpp)

```
    #include<iostream>
    #include<cmath>
    using namespace std;
    int main()
    {
      float a[50][50],b[50][50],C[50][50];
                int RowSize=50,ColSize=50;
      for(int i=0;i<RowSize;i++)
      {
                    for(int j=0;j<ColSize;j++)
                    {
              a[i][j]=0.0+i*0.5;
              b[i][j]=0.0+i*0.5;
                        }
      }
                for(int i=0;i<RowSize;i++)
                {
                  for(int j=0;j<ColSize;j++)
                  {
                      C[i][j]=a[i][j]+b[i][j];
                  }
                }
//Print
          return 0;
    }
```

Examples

Write a C++ program to multiply two matrices (MatrixMultiplication.cpp)

```
    //Fill the rest
float A[m][n], B[n][p], C[m][p]
for(int i=0;i<m;i++)
{
  for(int j=0;j<p;j++)
  {
    C[i][j]=0.0;
    for(int k=0;k<n;k++)
    {
      C[i][j]+=a[i][k]*b[k][j];
    }
  }
}

//Fill the rest
```

Common Mistakes

- Forgetting to initialize an array
- Referring to an element outside the array bounds
- Keeping array indices below 0 when looping or going beyond its size-1

Thanks

Doubts and Suggestions

panch.m@iittp.ac.in