[Example: 12] for Loop Example

```
//    Example of a for loop
//    Print number series from 1 to user limit.

#include <iostream.h>

int main (void)
{
//    Local Declarations
      int i;
      int limit;

//    Statements
      cout << "\nPlease enter a limit: ";
      cin  >> limit;
      for (i = 1; i <= limit; i++)
          cout << "\t" << i << endl;

      return 0;
}     // main

/*
Results:
      Please enter a limit: 3
          1
          2
          3
*/
```


[Example: 13] do … while Loop Example

```
//    Adding a list with the do...while
//    Adds a list of integers from the standard input unit

#include <iostream.h>

int main (void)
{
//    Local Declarations
      int num;
      int sum;

      // Statements
      num = sum = 0;
      cout << "Enter your numbers: <EOF> to stop.\n";
      cin >> num;
      do
      {
            sum += num;
      } while (cin >> num);
      cout << "Total: " << sum << endl;
      return  0;
}     // main

/*
Results:
      Run 1:
      Enter your numbers: <EOF> to stop.
      10 15 20 25
      Total: 70

      Run 2:
      Enter your numbers: <EOF> to stop.
      Total: 0
*/
```

[Example: 14] for Loops Example: Compound Interest

```
//    Compound interest
//    Print report showing value of investment.

#include <iostream.h>
#include <iomanip.h>

int main (void)
{
//    Local declarations
      double presVal;
      double futureVal;
      double rate;

      int years;
      int looper;

//    Statements
      cout << "Enter value of investment:   ";
      cin  >> presVal;
      cout << "Enter rate of return (nn.n): ";
      cin  >> rate;
      cout << "Enter number of years:       ";
      cin  >> years;

      cout << "\nYear\t   Value\n";
      cout <<   "====\t========\n";

      cout.setf (ios::fixed);
      cout.setf (ios::showpoint);
      cout.precision(2);
```



```
      for (futureVal = presVal, looper = 1; looper <= years; looper++ )
      {
            futureVal = futureVal * (1 + rate/100.0 );
            cout << setw(3) << looper    << " \t";
            cout << setw(8) << futureVal << endl;
      } // for
      return 0;
}     // main

/*
Results:
      Enter value of investment:   10000
      Enter rate of return (nn.n): 7.2
      Enter number of years:       5

      Year       Value
      ====     ========
        1      10720.00
        2      11491.84
        3      12319.25
        4      13206.24
        5      14157.09
*/
```

[Example: 15] for Loops Example: Right Triangle

```
//    Print right triangle using nested for loops
//    Print a number series from 1 to a user specified limit in the form of a right triangle.

#include <iostream.h>

int main (void)
{
//    Local Declarations
      int lineCtrl;
      int numCtrl;
      int limit;

//    Statements
      // Read limit
      cout << "\nPlease enter a number between 1 and 9: ";
      cin  >> limit;

      for (lineCtrl = 1; lineCtrl <= limit; lineCtrl++ )
      {
            for (numCtrl = 1; numCtrl <= lineCtrl; numCtrl++)
              cout << numCtrl;
            cout << endl;
      } // for lineCtrl

      return 0;
}     // main

/* Results
      Please enter a number between 1 and 9: 4
      1
      12
      123
      1234
*/
```


[Example: 16] for Loops Example: Print Rectangle

```
//    Print rectangle using nested for loops
//    Print number series 1 to a user specified limit in the form of a rectangle.

#include <iostream.h>
int main (void)
{
//    Local Declarations
      int row;
      int col;
      int limit;
//    Statements
      // Read limit
      cout << "\nEnter a number between 1 and 9: ";
      cin  >> limit;
      for (row = 1; row <= limit; row++)
      {
            for  (col = 1; col <= limit; col++)
                  if (row >= col)
                        cout << col;
                  else
                        cout << '*';
                  cout << endl;
      } // for row ...
      return 0;
}     // main

/* Results

      Please enter a number between 1 and 9: 5
      1****
      12***
      123**
      1234*
      12345
*/
```

[Example: 17] while Loops Example: Print Sum of Digits

```
//    Print sum of digits
//    Print the number and sum of digits in an integer.
#include <iostream.h>
#include <iomanip.h>
int main (void)
{
//    Local Declarations
      int number;
      int count;
      int sum;
//    Statements
      count = 0;
      sum   = 0;
      cout << "\nEnter an integer: ";
      cin  >> number;
      cout << "Your number is:   " << number << endl;
      while (number != 0)
      {
            count++;
            sum +=  number % 10;
            number /= 10;
      } // while
      cout << "\nThe number of digits is : " << setw(3) << count << endl;
      cout << "The sum of the digits is: " << setw(3) << sum   << endl;
      return 0;
}     // main

/*
Results:
      Enter an integer: 12345
      Your number is:   12345

      The number of digits is:    5
      The sum of the digits is:  15
*/
```


[Example: 18] Array Example: Squares Array

```
//    Squares array
//    Initialize array with square of index and print it.

#include <iostream.h>
#include <iomanip.h>
const int ARY_SIZE = 5;
int main (void)
{
// Local Declarations
      int i;
      int sqrAry[ARY_SIZE];
// Statements
      for (i = 0; i < ARY_SIZE; i++)
            sqrAry[i] = i * i;
      cout << "Element\tSquare\n";
      cout << "=======\t======\n";
      for (i = 0; i < ARY_SIZE; i++)
      {
            cout << setw(5) << i << "\t";
            cout << setw(5) << sqrAry[i] << endl;
      } // for i
      return 0 ;
}     /* main */

/*
Results:

Element Square
======= ======
    0       0
    1       1
    2       4
    3       9
    4      16
*/
```

[Example: 19] Two – Dimensional Array

```
//    two-dimensional array

#include <iostream.h>
#include <iomanip.h>

const int ROWS = 4;
const int COLS = 5;

int main (void)
{
//    Local Declarations
      int table [ROWS] [COLS] =
                  {
                        { 00, 01, 02, 03, 04 },
                        { 10, 11, 12, 13, 14 },
                        { 20, 21, 22, 23, 24 },
                        { 30, 31, 32, 33, 34 },
                  };
      int line [ROWS * COLS];
      int row;
      int column;

//    Statements
      for (row = 0; row < ROWS; row++)
      {
            for (column = 0; column < COLS; column++)
            {
                  cout << table[row][column] << "  " ;
            }
            cout << "\n" << endl;
      }

      return 0;
}     // main
```

```
/*
Results:
      0   1   2   3   4
      10  11  12  13  14
      20  21  22  23  24
      30  31  32  33  34
      Press any key to continue
*/
```


[Example: 20] Selection Sort

```
//    selectionSort
//    Test driver for sorting.

#include <iostream.h>
#include <iomanip.h>

const int MAX_ARY_SIZE = 15;

/*    Prototype Declarations */
      void selectionSort    (int list[ ], int last);
      void exchangeSmallest (int list[], int first, int last);

int main (void)
{
/*    Local Declarations */
      int i;
      int   ary[ MAX_ARY_SIZE ] = {89,72,3,15,21,57,61,44,19,98,5,77,39,59,61};

/*    Statements */
      cout << "Unsorted array: ";
      for (i = 0; i < MAX_ARY_SIZE; i++)
            cout << setw(3) << ary[i];

      selectionSort (ary, MAX_ARY_SIZE - 1);

      cout << "\nSorted array:   ";
      for (i = 0; i < MAX_ARY_SIZE; i++)
            cout << setw(3) << ary[i];
      cout << endl;
      return 0;
}     /* main */
```

```
/*
Results:
      Unsorted array:  89 72  3 15 21 57 61 44 19 98  5 77 39 59 61
      Sorted array:     3  5 15 19 21 39 44 57 59 61 61 72 77 89 98
*/
```

```
/*   Sorts by selecting smallest element in unsorted portion of array and exchanging it with
     element at the beginning of the unsorted list.
     Pre   list must contain at least one item.
           last contains index to last element in the list
     Post  The list has been rearranged smallest to largest
*/
void selectionSort (int list[ ], int last)
{
//   Local Declarations
     int current;
//   Statements
     for (current = 0; current < last; current++)
        exchangeSmallest (list, current, last);
     return;
}    // selectionSort

void exchangeSmallest (int list[ ], int current, int last )
/*   Given array of integers, place smallest element into  position in array.
     Pre:  list must contain at least one element current is beginning of array
           (not necessarily 0) last is last element in array. Must be >= current
     Post: returns index of smallest element in array.
*/
{    // exchangeSmallest
//   Local Declarations
     int walker ;
     int smallest ;
     int tempData ;
//   Statements
     smallest = current;
     for (walker = current + 1 ; walker <= last ; walker++)
        if ( list[ walker ] < list[ smallest ] )
           smallest = walker ;
     // Smallest selected: exchange with current element
      tempData       = list[ current ] ;
      list[current]  = list[ smallest ] ;
      list[smallest] = tempData ;
     return ;
}    // exchangeSmallest
```


[Example: 21] Bubble Sort

```
//   bubbleSort
//   Test driver for sorting.

#include <iostream.h>
#include <iomanip.h>

const int MAX_ARY_SIZE = 15;

//   Prototype Declarations
     void bubbleSort (int list [ ], int last);
     void bubbleUp (int list[ ], int first, int last);

int main (void)
{
//   Local Declarations
     int i;
     int   ary[ MAX_ARY_SIZE ] = {89,72,3,15,21,57,61,44,19,98,5,77,39,59,61};
//   Statements
     cout << "Unsorted array: ";
     for (i = 0; i < MAX_ARY_SIZE; i++)
          cout << setw(3) << ary[ i ];

     bubbleSort (ary, MAX_ARY_SIZE - 1);

     cout << "\nSorted array:   ";
     for (i = 0; i < MAX_ARY_SIZE; i++)
          cout << setw(3) << ary[ i ];
     cout << endl;
     return 0;
}    // main
     /*
     Results:
          Unsorted array:  89 72  3 15 21 57 61 44 19 98  5 77 39 59 61
          Sorted array:     3  5 15 19 21 39 44 57 59 61 61 72 77 89 98
     */
```

```
/*   Sort list using bubble sort. Adjacent elements are compared and exchanged until list is
     completely ordered.
     Pre  The list must contain at least one item.
          last contains index to last element in list.
     Post List has been rearranged in sequence low to high.
*/
void bubbleSort (int list [ ], int last)
{
//    Local Declarations
     int current;
//   Statements
      for(current = 0; current < last; current++)
         bubbleUp (list, current, last);
     return;
}    // bubbleSort

/*   Move the lowest element in unsorted portion of an array to the current element in the unsorted
     portion
     Pre    list must contain at least one element current identifies start of unsorted data
            last identifies end of the unsorted data
      Post  Array segment has been rearranged so that lowest element now at beginning of unsorted
            portion.
*/
void bubbleUp (int list[ ], int current, int last)
{
//   Local Declarations
     int walker;
     int temp;
//   Statements
     for (walker = last; walker > current; walker--)
          if (list[ walker ]  < list[ walker - 1 ])
          {
               temp              = list[walker];
               list[walker]      = list[walker - 1];
               list[walker - 1] = temp;
          } // if
     return;
}    // bubbleUp
```


[Example: 22] Insertion Sort

```
//   insertionSort
//   Test driver for sorting.

#include <iostream.h>
#include <iomanip.h>

const int MAX_ARY_SIZE = 15;

/*   Prototype Declarations */
     void  insertionSort (int list[ ], int last);
     void insertOne (int list[ ], int first);

int main (void)
{
/*   Local Declarations */
     int i;
     int   ary[ MAX_ARY_SIZE ] = {89,72,3,15,21,57,61,44,19,98,5,77,39,59,61};

/*   Statements */
     cout << "Unsorted array: ";
     for (i = 0; i < MAX_ARY_SIZE; i++)
          cout << setw(3) << ary[ i ];

     insertionSort (ary, MAX_ARY_SIZE - 1);

     cout << "\nSorted array:   ";
     for (i = 0; i < MAX_ARY_SIZE; i++)
          cout << setw(3) << ary[ i ];
     cout << endl;
     return 0;
}    /* main */
      /*
      Results:
           Unsorted array:  89 72  3 15 21 57 61 44 19 98  5 77 39 59 61
           Sorted array:     3  5 15 19 21 39 44 57 59 61 61 72 77 89 98
      */
```

```
/*   Sort list using Insertion Sort. The list is divided into sorted and unsorted list. With
     each pass, first element in unsorted list is inserted into sorted list.
     Pre   List must contain at least one element.
           last contains index to last element in the list
     Post  List has been rearranged.  */
void insertionSort (int list[ ], int last)
{
//   Local Declarations
     int current;
//   Statements
     for (current = 1; current <= last; current++)
        insertOne(list, current);
     return;
}    // insertionSort
/*   Sorts current element in unsorted list into its proper location in sorted portion of the
     list--one sort pass.
     Pre   list must contain at least one element
           current identifies beginning of unsorted list.
     Post  next element placed into its proper location  */
void insertOne (int list[ ], int current)
{
//   Local Declarations
     bool located;
     int  walker;
     int  temp;
//   Statements
     located = false;
     temp = list[current];
     for  (walker = current - 1; walker >= 0 && !located;)
          if (temp < list[walker])
          {    list[walker + 1] = list[walker];
               walker--;
          } // if
          else
               located = true;
     list [walker + 1] = temp;
     return;
}    // insertOne
```


[Example: 23] Sequential Search

```
//   Sequential search
//   This program tests searches.

#include <iostream.h>
#include <iomanip.h>

const int SIZE = 9;

/* Prototype Declarations */
int seqSearch (int list[ ], int last, int target, int& locn);

int main (void)
{
     /* Local Declarations */
     int i;
     int locn;
     int ary[ SIZE ] = {19, 13, 58, 10, 14, 1, 21, 64, 9};
     /* Statements */
     cout << "\nIndexes  : ";
     for (i = 0; i < SIZE; i++)
          cout << setw(3) << i;
     cout << "\nList data: ";
     for (i = 0; i < SIZE; i++)
          cout << setw(3) << ary[i];
     cout << endl;

     if (seqSearch (ary, SIZE - 1,  9, locn))
          cout << "Found  9 at location " << locn << endl;
     else
          cout << " 9 Not Found at      " << locn << endl;

     if (seqSearch (ary, SIZE - 1, 19, locn))
          cout << "Found 19 at location " << locn << endl;
     else
          cout << "19 Not Found at      " << locn << endl;
```

```
    if (seqSearch (ary, SIZE - 1, 10, locn))
        cout << "Found 10 at location " << locn << endl;
    else
        cout << "10 Not Found at      " << locn << endl;

    if (seqSearch (ary, SIZE - 1, 0, locn))
        cout << "Found 0 at location  " << locn << endl;
    else
        cout << "0 Not Found at       " << locn << endl;

    cout << endl;

    return 0;
}  /* main */

/*  Locate the target in an unordered list of size elements.
    Pre:  list must contain at least one item.
          last contains index to last element in list
          target contains the data to be located
    Post: FOUND: matching index stored in locn address
                 return 1 (found)
          NOT FOUND: last stored in locn address.
                     return 0 (not found)
*/
int seqSearch (int  list[ ], int  last, int  target, int& locn)
{
//  Local Declarations
    int looker;

//  Statements
    looker = 0;
    while (looker < last && target != list[looker])
       looker++;

    locn = looker;
    return (target == list[looker]);
}   // seqSearch
/*  ================ End of Program ================ */
```



```
/*
Results:

    Indexes  :   0  1  2  3  4  5  6  7  8
    List data:  19 13 58 10 14  1 21 64  9
    Found  9 at location 8
    Found 19 at location 0
    Found 10 at location 3
    0 Not Found at       8

*/
```

[Example: 24] Binary Search

```
//    Binary search
//    This program tests the binary search.

#include <iostream.h>
#include <iomanip.h>

const int SIZE = 9;

//    Prototype Declarations
      bool binarySearch (int list[ ], int end, int target, int& locn);

int main (void)
{

//    Local Declarations
      int i;
      int locn;
      int ary[ SIZE ] = {1, 9, 10, 13, 14, 19, 21, 58, 64};

//    Statements
      cout << "\nIndexes  : ";
      for (i = 0; i < SIZE; i++)
            cout << setw(3) << i;

      cout << "\nList data: ";
      for (i = 0; i < SIZE; i++)
            cout << setw(3) << ary[i];
      cout << endl;

      if (binarySearch (ary, SIZE - 1,  1, locn))
            cout << "Found  1 at location " << locn << endl;
      else
            cout << " 1 Not Found at      " << locn << endl;
```



```
      if (binarySearch (ary, SIZE - 1,  9, locn))
            cout << "Found  9 at location " << locn << endl;
      else
            cout << " 9 Not Found at      " << locn << endl;

      if (binarySearch (ary, SIZE - 1, 64, locn))
            cout << "Found 64 at location " << locn << endl;
      else
            cout << "64 Not Found at      " << locn << endl;

      if (binarySearch (ary, SIZE - 1, -4, locn))
            cout << "Found -4 at location " << locn << endl;
      else
            cout << "-4 Not Found at      " << locn << endl;

      if (binarySearch (ary, SIZE - 1, 33, locn))
            cout << "Found 33 at location " << locn << endl;
      else
            cout << "33 Not Found at      " << locn << endl;

      if (binarySearch (ary, SIZE - 1, 93, locn))
            cout << "Found 93 at location " << locn << endl;
      else
            cout << "93 Not Found at      " << locn << endl;

       return 0;
}  // main
/*
Results:
      Indexes  :   0  1  2  3  4  5  6  7  8
      List data:   1  9 10 13 14 19 21 58 64
      Found  1 at location 0
      Found  9 at location 1
      Found 64 at location 8
      -4 Not Found at      0
      33 Not Found at      7
      93 Not Found at      8
*/
```

```
/*   Search an ordered list using Binary Search
     Pre  list must contain at least one element
          end--index to the largest element in list
          target is value of element being sought
     Post  FOUND--locn assigned index to target element
                return true
           NOT FOUND--locn = element below or above target
                return false
*/
bool binarySearch (int  list[ ], int  end, int  target, int& locn)
{
//   Local Declarations
     int first;
     int mid;
     int last;

//   Statements
     first = 0;
     last = end;
     while (first <= last)
     {
          mid = (first + last) / 2;
          if (target > list[ mid ])
               // look in upper half
               first = mid + 1;
          else if (target < list[ mid ])
               // look in lower half
               last = mid - 1;
          else
               // found equal: force exit
               first = last + 1;
     } // end while
     locn = mid;
     return target == list [mid];
}    // binarySearch

//   ================ End of Program ================
```


[Example: 25] Demonstrate use of pointers

```
//   Demonstrate use of pointers
//   Demonstrate pointer use

#include <iostream.h>

int main (void)
{
//   Local declarations
     int  a;
     int *p;

//   Statements
     a = 14;
     p = &a;

     cout << a << " " << &a << endl;
     cout << p << " " << *p << " " << a << endl;

     return 0;
}    // main

/*
Results:
     14 0x00d50a2a
     0x00d50a2a 14 14
*/
```

[Example: 26] Example: Variables and Fun with Pointers

```
//    Fun with pointers
#include <iostream.h>
int main (void)
{
//    Local Declarations
      int  a;
      int  b;
      int  c;
      int *p;
      int *q;
      int *r;
//    Statements
      a = 6;
      b = 2;
      p = &b;
      q = p;
      r = &c;

      p  = &a;
      *q = 8;
      c  = *q;

      *r = *p;
      *r = a + *q + *&c;

      cout <<  a << " " <<  b << " " <<  c << endl;
      cout << *p << " " << *q << " " << *r << endl;
      return 0;
}     // main

/*
Results:
      6 8 20
      6 8 20
*/
```


[Example: 27] Example: Pointer Flexibility

```
//    Using one pointer for many variables
//    This program shows how the same pointer can point to different data variables in different
//    statements.

#include <iostream.h>
#include <iomanip.h>

int main ( void )
{
//    Local Declarations
      int  a;
      int  b;
      int  c;
      int *p;

//    Statements
      cout << "\nEnter three numbers and key return: ";
      cin  >> a >> b >> c;

      p = &a;
      cout << setw(3) << *p << endl;
      p = &b;
      cout << setw(3) << *p << endl;
      p = &c;
      cout << setw(3) << *p << endl;
      return 0;
}     // main

/*
Results:
      Enter three numbers and key return: 5 8 -3
       5
       8
      -3
*/
```

[Example: 28] Example: Multiple Pointers for One Variable

```
//    Using one variable with many pointers
//    This program shows how we can use different pointers to point to the same data variable.

#include <iostream.h>

int main (void)
{
//    Local Declarations
      int  a;
      int *p = &a;
      int *q = &a;
      int *r = &a;

//    Statements
      cout << "\nEnter a number: ";
      cin  >> a;

      cout << *p << endl;
      cout << *q << endl;
      cout << *r << endl;

      return 0;
}     // main

/*
Results:
      Enter a number: 17
      17
      17
      17
*/
```


[Example: 29] Pointer–To–Pointer Example

```
//    Using pointers to pointers
//    Show how pointer to pointer can be used by different input functions to read data to the
//    same variable.

#include <iostream.h>
int main (void)
{
//    Local Declarations
      int    a;
      int   *p;
      int  **q;
      int ***r;
//    Statements
      p = &a;
      q = &p;
      r = &q;

      cout << "Enter a number: ";
      cin  >> a;
      cout << "Your number is: " << a << endl;

      cout << "\nEnter a number: ";
      cin  >> *p;
      cout << "Your number is: " << a << endl;

      cout << "\nEnter a number: ";
      cin  >> **q;
      cout << "Your number is: " << a << endl;

      cout << "\nEnter a number: ";
      cin  >> ***r;
      cout << "Your number is: " << a << endl;

      return 0;
}     // main
```

```
/*
Results:
      Enter a number: 1
      Your number is: 1

      Enter a number: 2
      Your number is: 2

      Enter a number: 3
      Your number is: 3

      Enter a number: 4
      Your number is: 4
*/
```