Example 1 (C++) [triangle.cpp]

```
// triangle.cpp
// evaluates triangular numbers
#include <iostream>
using namespace std;

int main()
{
    int theNumber;
    int triangle(int);
    cout << "Enter a number: ";
    cin >> theNumber;
    int theAnswer = triangle(theNumber);
    cout << "Triangle=" << theAnswer << endl;
    return 0;
}  // end main()
//-------------------------------------------------------------
int triangle(int n)
{
    if(n==1)
        return 1;
    else
        return  ( n + triangle(n-1) );
}
```

Anagrams

cat,
    ,cta
atc,
    ,act
tca,
    ,tac

Example 2 (C++) [anagrams.cpp]

```
#include <iostream>
#include <string>
#include <cstdlib>

using std::cin;   using std::cout;
using std::endl;  using std::string;

class word
{
    private:
        string slowo;
        int n;
        int how_many;

    public:
        word(string s):slowo(s),n(slowo.length()),how_many(0)
        {}

    void rotate(int size){
```

```
        int position=n-size;
        char temp=slowo[position];
        for(int j=position+1;j<n;j++){
            slowo[j-1]=slowo[j];
        }
        slowo[n-1]=temp;
    }

    void anagram(int size){
        if(size==1)
            return;

        for(int j=0;j<size;j++){
            anagram(size-1);
            if(size==2){
                cout<<slowo<<" ";
                how_many++;
                if(how_many%6==0){
                    cout<<endl;
                }
            }
            rotate(size);
        }
    }
};
int main(){
    cout<<"Wpisz slowo: ";
    string ts;
    cin>>ts;
    int dlugoscslowa=ts.length();

    word s(ts);
    s.anagram(dlugoscslowa);

    system("pause");
    return 0;
}
```

Binary Search

Example 3 (C++). [BinarySearch.cpp]

```
#include <iostream>
#include <cstdlib>

using namespace std;
int t[12]={1,2,6,18,20,23,29,32,34,39,40,41};

void bs(int * m, int szuk, int left, int right){
    int mid=(left+right)/2;
    cout<<mid<<" ("<<m[mid]<<"),"<<left<<" "<<right<<endl;
    if(find==m[mid]){
            cout<<"An item "<<find<<" has been found.\n***************\n";
            return;
    }
    else{
        if(find!=m[mid]&&right==left){
                cout<<"An item "<<find<<" has not been found.\n";
```

```
        }
    }
    if(right>left){
        if(find<m[mid]){
                right=mid-1;
        }
        else{
            left=mid+1;
        }
        bs(m,find,left,right);
    }
}

int main(){

int s=0;
for(int i=1;i<12;i++){
    s=t[i];
    cout << "Finding of s="<<s<<" item in an array t[]:\n";
    bs(t,s,0,11);
}
cout<<"\n ################ \n";
for(int i=1;i<12;i++){
    s=t[i]+41;
    cout << "Finding of s="<<s<<" item in an array t[]:\n";
    bs(t,s,0,11);
}
    s=3;
    cout << "Finding of s="<<s<<" item in an array t[]:\n";
    bs(t,s,0,11);
cout<<endl;

system("pause");
return 0;
}
```

Merging of sorted arrays

Example 4 (C++) [merge.cpp]

```
//merge.cpp
//demonstrates merging two arrays into a third
#include <iostream>
using namespace std;

void merge( int[], int, int[], int, int[] );
void display(int[], int);               //prototypes

int main()
{
    int arrayA[] = {23, 47, 81, 95};        //source A
    int arrayB[] = {7, 14, 39, 55, 62, 74}; //source B
    int arrayC[10];                         //destination

    merge(arrayA, 4, arrayB, 6, arrayC);    //merge A+B-->C
    display(arrayC, 10);                    //display result
    return 0;
}  //end main()
```

```
void merge( int arrayA[], int sizeA,        //merge A and B into C
int arrayB[], int sizeB,
int arrayC[] )
{
    int aDex=0, bDex=0, cDex=0;
    while(aDex < sizeA && bDex < sizeB)     //neither array empty
        if( arrayA[aDex] < arrayB[bDex] )
            arrayC[cDex++] = arrayA[aDex++];
        else
            arrayC[cDex++] = arrayB[bDex++];
    while(aDex < sizeA)                     //arrayB is empty,
        arrayC[cDex++] = arrayA[aDex++];    //but arrayA isn't
    while(bDex < sizeB)                     //arrayA is empty,
        arrayC[cDex++] = arrayB[bDex++];    //but arrayB isn't
}  //end merge()

void display(int theArray[], int size)      //display array
{
    for(int j=0; j<size; j++)
    cout << theArray[j] << " ";
    cout << endl;
}
```

Sorting by merging

Example 5 (C++) [mergeSort.cpp]

```
//mergeSort.cpp
//demonstrates recursive merge sort
#include <iostream>
#include <vector>
using namespace std;

class DArray
{
    private:
        vector<double>(theVect);            //vector of doubles
        int nElems;                         //number of data items
        void recMergeSort(vector<double>, int, int);
        void merge(vector<double>, int, int, int);

    public:
        DArray(int max) : nElems(0)         //constructor
        {
            theVect.resize(max);            //size vector
        }

        void insert(double value)           //put element into array
        {
            theVect[nElems] = value;        //insert it
            nElems++;                       //increment size
        }

        void display()                      //displays array contents
        {
            for(int j=0; j<nElems; j++)     //for each element,
                cout << theVect[j] << " ";  //display it
```

```
            cout << endl;
        }

        void mergeSort()                    //called by main()
        {                                   //provides workspace
            vector<double>(workSpace);
            workSpace.resize(nElems);
            recMergeSort(workSpace, 0, nElems-1);
        }
};  //end class DArray
//-------------------------------------------------------------
void DArray::recMergeSort(vector<double> workSpace,
int lowerBound, int upperBound)
{
    if(lowerBound == upperBound)        //if range is 1,
        return;                         //no use sorting
    else
    {                                   //find midpoint
        int mid = (lowerBound+upperBound) / 2;
                                        //sort low half
        recMergeSort(workSpace, lowerBound, mid);
                                        //sort high half
        recMergeSort(workSpace, mid+1, upperBound);
                                        //merge them
        merge(workSpace, lowerBound, mid+1, upperBound);
    }  //end else
}  //end recMergeSort()

void DArray::merge(vector<double> workSpace, int lowPtr,
int highPtr, int upperBound)
{
    int j = 0;                          //workspace index
    int lowerBound = lowPtr;
    int mid = highPtr-1;
    int n = upperBound-lowerBound+1;    //# of items

    while(lowPtr <= mid && highPtr <= upperBound)
        if( theVect[lowPtr] < theVect[highPtr] )
            workSpace[j++] = theVect[lowPtr++];
        else
            workSpace[j++] = theVect[highPtr++];

    while(lowPtr <= mid)
        workSpace[j++] = theVect[lowPtr++];

    while(highPtr <= upperBound)
        workSpace[j++] = theVect[highPtr++];

    for(j=0; j<n; j++)
        theVect[lowerBound+j] = workSpace[j];
}  //end merge()

int main()
{
    const int maxSize = 100;            //array size
    DArray arr(maxSize);                //create "array"

    arr.insert(64);                     //insert items
    arr.insert(21);
    arr.insert(33);
    arr.insert(70);
```

```
    arr.insert(12);
    arr.insert(85);
    arr.insert(44);
    arr.insert(3);
    arr.insert(99);
    arr.insert(0);
    arr.insert(108);
    arr.insert(36);

    arr.display();                  //display items
    arr.mergeSort();                //merge-sort the array
    arr.display();                  //display items again
    return 0;
}  //end main()
```

Anagrams once more

Program no. 2 is enriched about the container that stores anagrams and calculate time of finding of these anagrams (changes highlighted by a yellow background).

Example 6 (C++) [anagrams2.cpp]

```
include <iostream>
#include <string>
#include <cstdlib>
#include <vector>
#include <ctime>

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

class word
{
    private:
        string word;
        int n;
        int how_many;
        vector<string> anagrams;

    public:
        word(string s):word(s),n(word.length()),how_many(0)
        {}

    void rotate(int size){
        int position=n-size;
        char temp=word[position];
        for(int j=position+1;j<n;j++){
            word[j-1]=word[j];
        }
        word[n-1]=temp;
    }

    void anagram(int size){
        if(size==1)
```

```
            return;

        for(int j=0;j<size;j++){
            anagram(size-1);
            if(size==2){
                anagrams.push_back(word);
                how_many++;
            }
            rotate(size);
        }
    }

    int how_many(){
        return how_many;
    }

    void display(){
        for(unsigned int i=0;i<anagrams.size();i++){
            cout<<anagrams[i]<<" ";
            if(i%6==0){
                cout<<endl;
            }
        }
    }
};


int main(){
    clock_t start1, finish1;
    clock_t clock(void);  //Returns the processor time consumed by the
                          // program
                          //clock_t is the type returned by clock.
    double duration1;


    cout<<"Enter a word: ";
    string ts;
    cin>>ts;
    int length_of_the_word=ts.length();

    word s(ts);

    start1 = clock();
    s.anagram(length_of_the_word);
    finish1 = clock();
    duration1 = static_cast<double>(finish1 - start1)/CLOCKS_PER_SEC;

    //s.display();          //displaying takes extra time

    cout<<endl<<"Found "<<s.how_many()<<" anagrams of the word \'"<<ts
    <<"\' in "<<duration1<<" seconds"<<endl;


    system("pause");
    return 0;
}
```

Based on:
Robert Lafore – "Teach Yourself Data Structures And Algorithms In 24 Hours"