1. Classes and objects, methods.
2. Arrays.

Task 1 (C++) Creating new objects. Concepts related to development of new objects.

[t1.cpp]

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

using namespace std;

//keyword 'class'
class dog
{
    //private - you can get access only through member function(s)
    private:            //keyword 'private'
    //data (member) variable(s):
        string name;    //member variable (of type string)
        double weight;  //member variable (of type double)

    public:             //keyword 'public'
        dog() :         //Default constructor (Default C-tor)
        name("Max"), weight(5)      //initializer list
        { cout << "Default constructor" << endl; }

        //C-tor with parameters
        dog(string i, double w=10)
        {
            name=i;
            weight=w;
            cout << "Constructor with parameters"<<endl;
        }

        //member function(s) -  method(s):
        void info(){
            cout<<"name: "<<name<<", weight: "<< weight <<"kg"<<endl;
        }
};//semicolon!

int main()
{
    int a;      //a is an object of type int
    a=20;

    dog x;     //x is an object/instance of type/class dog
    x.info();  // calling a method for an object x

    x=dog("Murphy",a);
    x.info();

    dog y("Lucky");
    y.info();

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

An array is a fragment of a memory with a specific number of elements. Below we will see how to manage arrays (inserting a new key value to an array, delete the item containing a key value, finding item with specified key).

Task 2 (C++) An example of a procedural program for arrays.

[array.cpp]

```
//array.cpp
//demonstrates arrays
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////////////
int main()
{
    int arr[100];               //array
    int nElems = 0;             //number of items
    int j;                      //loop counter
    int searchKey;              //key of item to search for
//-----------------------------------------------------------
    arr[0] = 77;                //insert 10 items
    arr[1] = 99;
    arr[2] = 44;
    arr[3] = 55;
    arr[4] = 22;
    arr[5] = 88;
    arr[6] = 11;
    arr[7] = 00;
    arr[8] = 66;
    arr[9] = 33;
    nElems = 10;                //now 10 items in array
//-----------------------------------------------------------
    for(j=0; j<nElems; j++)     //display items
        cout << arr[j] << " ";
    cout << endl;
//-----------------------------------------------------------
    searchKey = 66;             //find item with key 66
    for(j=0; j<nElems; j++)     //for each element,
        if(arr[j] == searchKey) //found item?
            break;              //yes, exit before end
    if(j == nElems)             //at the end?
        cout << "Can't find " << searchKey << endl; //yes
    else
        cout << "Found " << searchKey << endl;      //no
//-----------------------------------------------------------
    searchKey = 55;             //delete item with key 55
    cout << "Deleting " << searchKey << endl;
    for(j=0; j<nElems; j++)     //look for it
        if(arr[j] == searchKey)
            break;
    for(int k=j; k<nElems; k++) //move higher ones down
        arr[k] = arr[k+1];
    nElems--;                   //decrement size
//-----------------------------------------------------------
    for(j=0; j<nElems; j++)     //display items
        cout << arr[j] << " ";
```

```
    cout << endl;
    return 0;
}  //end main()
```

Inserting new an item.
Inserting into an array is simple:
arr[0] = 77;

Searching an item.
Variable searchKey stores the searched value. In order to find item such as searchKey, we need to look through the entire array step by step. Once the variable j has reached the last cell without hitting it means that search value is not in the array.

Deleting an item.
Erasing starts with finding specified item. If the item with a specific search value occurs in the array we need to move all the items of index greater than erased down one item. Then we reduce the value of the variable nElems.

Viewing the contents of the array.
The course of the entire array by displaying the value of each array element.

Gradually, in the next two examples, using an arrays we will be approached in an object-oriented way.

Task 3 (C++) First approach.

[lowArray.cpp]

```
//lowArray.cpp
//demonstrates array class with low-level interface
#include <iostream>
#include <vector>
using namespace std;
////////////////////////////////////////////////////////////////
class LowArray
{
    private:
        vector<double> v;                   //vector holds doubles

    public:
//--------------------------------------------------------------
    LowArray(int max)                       //constructor
    { v.resize(max); }                      //size the vector
//--------------------------------------------------------------
    void setElem(int index, double value)   //put element into
    { v[index] = value; }                   //array, at index
//--------------------------------------------------------------
    double getElem(int index)               //get element from
    { return v[index]; }                    //array, at index
    //--------------------------------------------------------------
};  //end class LowArray
////////////////////////////////////////////////////////////////
int main()
{
```

```
    LowArray arr(100);                      //create a LowArray
    int nElems = 0;                         //number of items
    int j;                                  //loop variable
//--------------------------------------------------------------
    arr.setElem(0, 77);                     //insert 10 items
    arr.setElem(1, 99);
    arr.setElem(2, 44);
    arr.setElem(3, 55);
    arr.setElem(4, 22);
    arr.setElem(5, 88);
    arr.setElem(6, 11);
    arr.setElem(7, 00);
    arr.setElem(8, 66);
    arr.setElem(9, 33);
    nElems = 10;                            //now 10 items in array
//--------------------------------------------------------------
    for(j=0; j<nElems; j++)                 //display items
        cout << arr.getElem(j) << " ";
    cout << endl;
//--------------------------------------------------------------
    int searchKey = 26;                     //search for item
    for(j=0; j<nElems; j++)                 //for each element,
        if(arr.getElem(j) == searchKey)         //found item?
            break;
    if(j == nElems)                         //no
        cout << "Can't find " << searchKey << endl;
    else                                    //yes
        cout << "Found " << searchKey << endl;
//--------------------------------------------------------------
    double deleteKey = 55;                  //delete value 55
    cout << "Deleting element " << deleteKey << endl;
    for(j=0; j<nElems; j++)                 //look for it
        if(arr.getElem(j) == deleteKey)
            break;
    for(int k=j; k<nElems; k++)             //higher ones down
        arr.setElem(k, arr.getElem(k+1) );
    nElems--;                               //decrement size
//--------------------------------------------------------------
    for(j=0; j<nElems; j++)                 //display items
        cout << arr.getElem(j) << " ";
    cout << endl;
    return 0;
}  //end main()
```

Instead of arrays we used class vector. In many aspects, the operations on objects such as vector are similar into operations on the arrays. Use of class vector allows us to determine the size of the vector (this is in a class constructor lowArray using the method resize()). It allows to store any amount of data as opposed to arrays which were restricted into the same size of an array.

Often structure to store data (vector) will be called simply array.

In the class lowArray array is hidden from the outside world. Only member functions of class lowArray can get access to an array (private). These functions are: setElem(), getElem() and a constructor.

Task 4 (C++) Second approach.

[highArray.cpp]

```
//highArray.cpp
//demonstrates array class with high-level interface
#include <iostream>
#include <vector>
using namespace std;
////////////////////////////////////////////////////////////////
class HighArray
{
    private:
        vector<double> v;                 //vector v
        int nElems;                       //number of data items
    public:
//--------------------------------------------------------------
    HighArray() : nElems(0)               //default constructor
        {  }
//--------------------------------------------------------------
    HighArray(int max) : nElems(0)        //1-arg constructor
        { v.resize(max); }                //size the vector
//--------------------------------------------------------------
    bool find(double searchKey)           //find specified value
    {
        int j;
        for(j=0; j<nElems; j++)           //for each element,
            if(v[j] == searchKey)         //found item?
                break;                    //exit loop before end
        if(j == nElems)                   //gone to end?
            return false;                 //yes, can't find it
        else
            return true;                  //no, found it
    }  //end find()
//--------------------------------------------------------------
    void insert(double value)             //put element into array
    {
        v[nElems] = value;                //insert it
        nElems++;                         //increment size
    }
//--------------------------------------------------------------
    bool remove(double value)             //remove element from array
    {
        int j;
        for(j=0; j<nElems; j++)           //look for it
            if( value == v[j] )
                break;
        if(j==nElems)                     //can't find it
            return false;
        else                              //found it
        {
            for(int k=j; k<nElems; k++)   //move higher ones down
                v[k] = v[k+1];
            nElems--;                     //decrement size
            return true;
        }
    }  //end delete()
//--------------------------------------------------------------
    void display()                        //displays array contents
```

```
    {
    for(int j=0; j<nElems; j++)          //for each element,
        cout << v[j] << " ";             //display it
    cout << endl;
    }
//--------------------------------------------------------------
};  //end class HighArray
////////////////////////////////////////////////////////////////
int main()
{
    int maxSize = 100;                   //array size
    HighArray arr(maxSize);              //vector
    arr.insert(77);                      //insert 10 items
    arr.insert(99);
    arr.insert(44);
    arr.insert(55);
    arr.insert(22);
    arr.insert(88);
    arr.insert(11);
    arr.insert(0);
    arr.insert(66);
    arr.insert(33);

    arr.display();                       //display items

    int searchKey = 35;                  //search for item
    if( arr.find(searchKey) )
        cout << "Found " << searchKey << endl;
    else
        cout << "Can't find " << searchKey << endl;

    cout << "Deleting 0, 55, and 99" << endl;
    arr.remove(0);                       //delete 3 items
    arr.remove(55);
    arr.remove(99);

    arr.display();                       //display items again
    return 0;
}  //end main()
```

User of class (ie. main()) does not have to keep an eye on the indexing table. Moreover, the user class does not need to know anything about the structure that stores data used in highArray. This structure is hidden behind the interface. In the next example, we'll see the same interface with a different data structure.

Abstraction

The separation process *how* from *what* (*how* operations are performed in the class, in contrast to *what* is seen by the user class) is called abstraction.

Task 5 (C++) An example of a class with an array containing ordered data. In this example, to element identification has been applied binary search (only ordered data).

[orderedArray.cpp]

```
//orderedArray.cpp
//demonstrates ordered array class
#include <iostream>
#include <vector>
using namespace std;
////////////////////////////////////////////////////////////////
class OrdArray
{
    private:
        vector<double> v;                 //vector v
        int nElems;                       //number of data items
    public:
//-----------------------------------------------------------
    OrdArray(int max) : nElems(0)         //constructor
    { v.resize(max); }                    //size the vector
//-----------------------------------------------------------
    int getSize()                         //return number of
    { return nElems; }                    //elements
//-----------------------------------------------------------
    int find(double searchKey)
    {
        int lowerBound = 0;
        int upperBound = nElems-1;
        int curIn;

        while(true)
        {
            curIn = (lowerBound + upperBound ) / 2;
            if(v[curIn]==searchKey)
                return curIn;                 //found it
            else if(lowerBound > upperBound)
                return nElems;                //can't find it
            else                              //divide range
            {
                if(v[curIn] < searchKey)
                    lowerBound = curIn + 1; //it's in upper half
                else
                    upperBound = curIn - 1; //it's in lower half
            }  //end else divide range
        }  //end while
    }  //end find()
//-----------------------------------------------------------
    void insert(double value)             //put element into array
    {
        int j;
        for(j=0; j<nElems; j++)           //find where it goes
            if(v[j] > value)              //(linear search)
                break;
        for(int k=nElems; k>j; k--)       //move bigger ones up
            v[k] = v[k-1];
        v[j] = value;                     //insert it
        nElems++;                         //increment size
    }  //end insert()
//-----------------------------------------------------------
    bool remove(double value)
```

```
    {
        int j = find(value);
        if(j==nElems)                           //can't find it
            return false;
        else                                    //found it
        {
            for(int k=j; k<nElems; k++)         //move bigger ones down
                v[k] = v[k+1];
            nElems--;                           //decrement size
            return true;
        }
    }  //end remove()
//-------------------------------------------------------------
    void display()                              //displays array contents
    {
        for(int j=0; j<nElems; j++)             //for each element,
            cout << v[j] << " ";                //display it
        cout << endl;
    }
//-------------------------------------------------------------
};  //end class OrdArray
////////////////////////////////////////////////////////////////
int main()
{
    int maxSize = 100;              //array size
    OrdArray arr(maxSize);          //create the array

    arr.insert(77);                 //insert 10 items
    arr.insert(99);
    arr.insert(44);
    arr.insert(55);
    arr.insert(22);
    arr.insert(88);
    arr.insert(11);
    arr.insert(00);
    arr.insert(66);
    arr.insert(33);

    int searchKey = 55;             //search for item

    if( arr.find(searchKey) != arr.getSize() )
        cout << "Found " << searchKey << endl;
    else
        cout << "Can't find " << searchKey << endl;

    arr.display();                  //display items

    cout << "Deleting 0, 55, and 99" << endl;
    arr.remove(00);                 //delete 3 items
    arr.remove(55);
    arr.remove(99);

    arr.display();                  //display items again
    return 0;
}  //end main()
```

Benefit of applying of ordered arrays

The main benefit is faster searching. However, we lose on the speed of inserting new data into an array (data with higher values must be moved up an array to make room for the new value).

Task 6 (C++) Object storage. Implementation of class Person. An example of a class that can be used for staff data storage.

[orderedArray.cpp]

```
//classDataArray.cpp
//data items as class objects
#include <iostream>
#include <string>
#include <vector>
using namespace std;
////////////////////////////////////////////////////////////////
class Person
{
    private:
        string lastName;
        string firstName;
        int age;
    public:
//-----------------------------------------------------------------
    Person(string last, string first, int a) :   //constructor
    lastName(last), firstName(first), age(a)
    {  }
//-----------------------------------------------------------------
    void displayPerson()
    {
        cout << "   Last name: " << lastName;
        cout << ", First name: " << firstName;
        cout << ", Age: " << age << endl;
    }
//-----------------------------------------------------------------
    string getLast()                    //get last name
    { return lastName; }
};  //end class Person
////////////////////////////////////////////////////////////////
class ClassDataArray
{
    private:
        vector<Person*> v;               //vector of pointers
        int nElems;                      //number of data items
    public:
//-----------------------------------------------------------------
    ClassDataArray(int max) : nElems(0) //constructor
    { v.resize(max); }                  //create the array
//-----------------------------------------------------------------
    ~ClassDataArray()                   //destructor
    {
        for(int j=0; j<nElems; j++)     //delete each element
            delete v[j];
    }
//-----------------------------------------------------------------
    Person* find(string searchName)
    {                                   //find specified value
        int j;
        for(j=0; j<nElems; j++)         //for each element,
            if( v[j]->getLast() == searchName )  //found item?
                break;                  //exit loop before end
        if(j == nElems)                 //gone to end?
            return NULL;                //yes, can't find it
        else
```

```
            return v[j];                    //no, found it
    };  //end find()
//--------------------------------------------------------------
    //put person into array
    void insert(string last, string first, int age)
    {
        v[nElems] = new Person(last, first, age);
        nElems++;                           //increment size
    }
//--------------------------------------------------------------
    bool remove(string searchName)      //delete person from array
    {
        int j;
        for(j=0; j<nElems; j++)         //look for it
            if( v[j]->getLast() == searchName )
                break;
        if(j==nElems)                   //can't find it
            return false;
        else                            //found it
        {
            delete v[j];                //delete Person object
            for(int k=j; k<nElems; k++) //shift down
                v[k] = v[k+1];
            nElems--;                   //decrement size
            return true;
        }
    }  //end remove()
//--------------------------------------------------------------
    void displayA()                     //displays array contents
    {
        for(int j=0; j<nElems; j++)     //for each element,
            v[j]->displayPerson();      //display it
    }
//--------------------------------------------------------------
};  //end class ClassDataArray
////////////////////////////////////////////////////////////////
int main()
{
    int maxSize = 100;                  //array size
    ClassDataArray arr(maxSize);        //array

    arr.insert("Evans", "Patty", 24);   //insert 10 items
    arr.insert("Smith", "Lorraine", 37);
    arr.insert("Yee", "Tom", 43);
    arr.insert("Adams", "Henry", 63);
    arr.insert("Hashimoto", "Sato", 21);
    arr.insert("Stimson", "Henry", 29);
    arr.insert("Velasquez", "Jose", 72);
    arr.insert("Lamarque", "Henry", 54);
    arr.insert("Vang", "Minh", 22);
    arr.insert("Creswell", "Lucinda", 18);

    arr.displayA();                     //display items

    string searchKey = "Stimson";       //search for item
    cout << "Searching for Stimson" << endl;
    Person* found;
    found=arr.find(searchKey);
    if(found != NULL)
    {
        cout << "   Found ";
```

```
        found->displayPerson();
    }
    else
        cout << "   Can't find " << searchKey << endl;

    cout << "Deleting Smith, Yee, and Creswell" << endl;
    arr.remove("Smith");                //delete 3 items
    arr.remove("Yee");
    arr.remove("Creswell");

    arr.displayA();                     //display items again
    return 0;
}  //end main()
```

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