**Assignment 6**

**Title of Assignment**: Implement all the functions of a dictionary (ADT) using hashing. Data: Set of (key, value) pairs, Keys are mapped to values, Keys must be comparable, Keys must be unique Standard Operations: Insert(key, value), Find(key), Delete(key)

**Relevant Theory:**

An ***abstraction*** is a simplified description, or *specification,* of a system that focuses on some essential structure or behavior of a real-world or conceptual object. A *good* abstraction is one in which [information](http://ecomputernotes.com/fundamental/information-technology/what-do-you-mean-by-data-and-information) that is significant to the user is emphasized while details that are immaterial, at least for the moment, are suppressed. We use the principles of *information hiding* to encapsulate these details.An *abstract* [*data type*](http://ecomputernotes.com/java/data-type-variable-and-array/explain-data-types-in-java)is a programming language facility for organizing programs into modules using criteria that are based on the data structures of the program. The specification of the module should provide all information required for using the type, including the allowable values of the data and the effects of the operations. However, details about the implementation, such as data representations and algorithms for implementing the operations, are hidden within the module. This separation of specification from implementation is a key idea of abstract data types. Each module that defines an abstract data type may include both data declarations and subroutine definitions. The criteria for organizing the modules emphasize protecting the data structures from arbitrary manipulation- malicious or accidental-by other parts of the program. Languages that support abstract data types include scope rules that guarantee this locality by hiding the names of local data from all parts of the program outside the module that defines the abstract data type. The objective of organizing a program using abstract data types is to expedite program development and to simplify maintenance by imposing a certain kind of predictable and useful structure on the program.

## Definitions for Implementing an ADT with a Class.

Before we proceed, let's just get our terminology in order:

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **class** | An implementation of an ADT. |
| **class member** | A **datum** associated with the class, or  **a function** used in the class to implement one of the operations associated with the class.        These functions are also referred to as **methods** of the class. |
| **class object** | An instance of the class. |

lass members are either ***public***, meaning that they are accessible from outside the class, or ***private*** (the default), meaning that only the class can access them. Generally data is designated as being private (look up "information hiding" in any text) and the methods are designated as being public, so that they can be called from elsewhere.

Here's a simple example:

class Computer

{

private:

// It's not necessary to include this term since that is the default.

// However, it's good form to do so and clarifies your meaning.

int processorSpeed;

public:

void setSpeed(int);

int getSpeed() const;

};

## Implementation Details.

### Member Functions

A header file - **Filename.h** - is often used to contain the function prototypes and data members for the class. Another file, generally the same *filename* but with an extension of **.cpp**, contains the actual function definitions. You must remember to **#include "Filename.h"** in the Filename**.cpp** and in the calling program.

When you enter the code for the function in the **Filename.cpp** file, you must precede each member function name with the name of the class.

e.g.

void Computer::setSpeed(int p)

The double colon **::** is known as the **scope resolution operator**. It ties the function name to the name of the related class. This is necessary because the function might also exist in another class.

Going back to the example we were just looking at, you could have three files involved in the total program.

|  |  |
| --- | --- |
| Computer.h | The header file for the Computer class.  This contains the private data members and the function  prototypes for the operations members. |
| Computer.cpp | The function definition file for the Computer class.  Must contain:   **#include "Computer.h"** |
|  |  |
| CompServ.cpp | The main program that calls the class.  Must contain:   **#include "Computer.h"** |

### Your code should be in three files:

* Date.h
  + contains the class definition
* Date.cpp
  + includes "Date.h"
  + contains the functions for the class
* main.cpp
  + includes "Date.h"
  + tests the class

Implementation of all the functions of a dictionary (ADT) using hashing

**1:hash.h**

Include declaration of all classes and functions

Class Hashentry-definition

Class Hashmap -definition

Function Insert(key,search)

,Delete(key)

,Display

Find(key)

**2:.hash.cpp**

Include “hash.h”

Define all functions which is declared in hash file

**3:hash1.cpp**

Include “hash.h”

Execute int main() program

**Conclusion: Understood and performed ADT implementation of Hashing.**