Welcome to our program! Today, we will be discussing and explaining the classes and functions in the provided code. Let's dive right in.

First, we have the Student class. It represents a student and has a constructor that takes two optional parameters, f name and l name, to set the first name and last name of the student, respectively.

The Student class also has a to string function, which returns a string representation of the student's full name by concatenating the first name and last name.

The static member functions are not super relevant, but they allow us to generate random names for our students. I encourage you to explore the details at a later time.

Most importantly, we have the HashMap class. It represents a hashmap data structure and contains a vector of Pair objects, where each Pair stores a student, an ID, and a flag indicating if the slot is occupied.

The HashMap class has a constructor that takes an optional parameter n to set the size of the hashmap. It initializes the size, resizes the vector based on the provided size, and calculates the number of slots in the hashmap based on a formula.

The class also provides various functions for accessing and modifying the hashmap:

Insert; inserts a student with an ID into the hashmap. If a pair with the same ID already exists, it updates the student. It uses the hash() function to calculate the index based on the ID and handles collisions using linear probing.

Remove; removes a pair from the hashmap based on the ID. It uses linear probing to find the pair with the given ID and marks the slot as unoccupied.

Get; retrieves the student object associated with the given ID. It uses linear probing to find the pair with the matching ID and returns the associated student object.

get size; returns the current size of the hashmap.

to string; returns a string representation of the known hashmap values. It iterates over the hashmap and collects the string representations of the occupied pairs.

A static function of the HashMap class, generate 6 digit id, will be used to assist us.

Now let's take a look at the main function and see how the program works.

In the main function, we first seed the random number generator using the current time.

We then create an instance of the HashMap class with a size of 25. This hashmap will store student information.

Next, we enter a loop to generate and insert 24 random students into the hashmap. Each student is generated using the generate student function, and a 6-digit ID is generated using the generate 6 digit id function.

After inserting the random students, we create a new Student object named "John Doe" and assign it the ID 101010.

We then call the insert function of the hashmap to insert the "John Doe" student with the ID 101010.

Next, we display the string representation of the hashmap using the to string function to visualize the known hashmap values.

We can see that despite the student John Doe being inserted last, it is not necessarily in the last position. This is the result of the hash function at work.

Now, let's look at the details of the student with the ID 101010 using the get function and its associated to string function.

We can see that the returned value is equivalent to what we'd expect.

When we remove the value and attempt to access it again, we are returned an empty string as expected.

That's the end of the main function. The program demonstrates the usage of the hashmap by inserting and removing student information based on their IDs.

In total, the size of the vector is only n times 2 minus n divided by 2 despite allowing for a total of almost 900,000 different IDs.

Another benefit is that the hash function obscures the location of the given ID in the vector. This means that I cannot just add 1 to my ID and be rewarded with the information of the student stored immediately after me.

This example used a simple hashing function and storage object, but there are a lot more powerful use cases for this concept. Let me know what you create!

As always, navigate to this location for the source code to explore on your own.