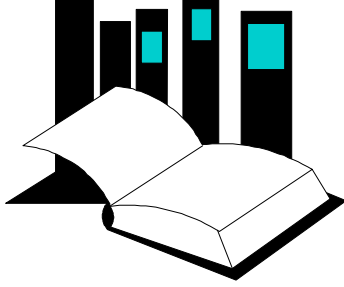
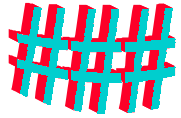



Hash Tables



**Data Structures
and Other Objects
Using C++**

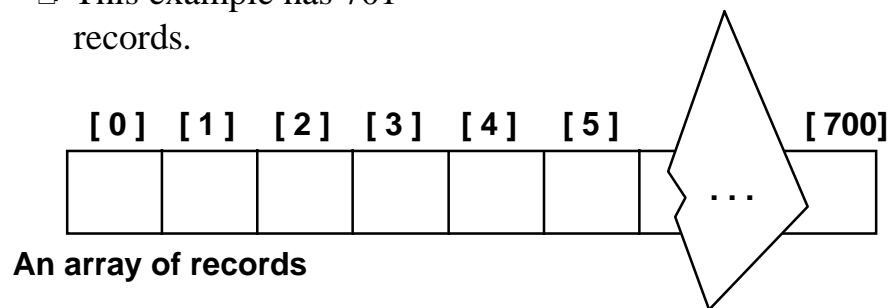
- ❑ Chapter 12 discusses several ways of storing information in an array, and later searching for the information.
- ❑ Hash tables are a common approach to the storing/searching problem.
- ❑ This presentation introduces hash tables.

This lecture illustrates hash tables, using open addressing.

Before this lecture, students should have seen other forms of a Dictionary, where a collection of data is stored, and each data item has a key associated with it.

What is a Hash Table ?

- The simplest kind of hash table is an array of records.
- This example has 701 records.

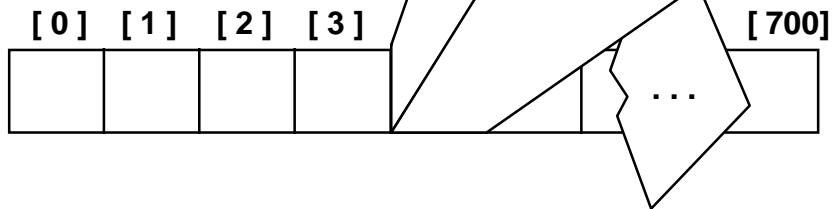


This lecture introduces hash tables, which are an array-based method for implementing a Dictionary. You should recall that we have seen dictionaries implemented in other ways, for example with a binary search tree. The abstract properties of a dictionary remain the same: We can insert items in the dictionary, and each item has a key associated with it. When we want to retrieve an item, we specify only the key, and the retrieval process finds the associated data.

What we do now is use an array to implement the dictionary. The array is an array of records. In this example, we could store up to 701 records in the array.

What is a Hash Table ?

- ❑ Each record has a special field, called its key.
- ❑ In this example, the key is a long integer field called Number.

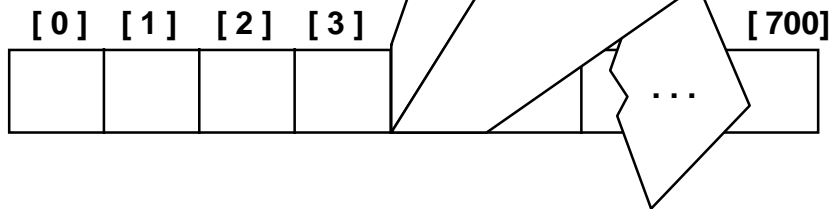


3

Each record in the array contains two parts. The first part is a number that we'll use for the key of the item. We could use something else for the keys, such as a string. But for a hash table, numbers make the most convenient keys.

What is a Hash Table ?

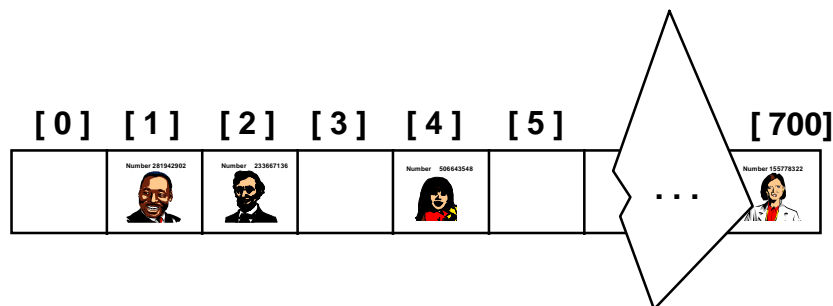
- The number might be a person's identification number, and the rest of the record has information about the person.



The numbers might be identification numbers of some sort, and the rest of the record contains information about a person. So the pattern that you see here is the same pattern that you've seen in other dictionaries: Each entry in the dictionary has a key (in this case an identifying number) and some associated data.

What is a Hash Table ?

- When a hash table is in use, some spots contain valid records, and other spots are "empty".



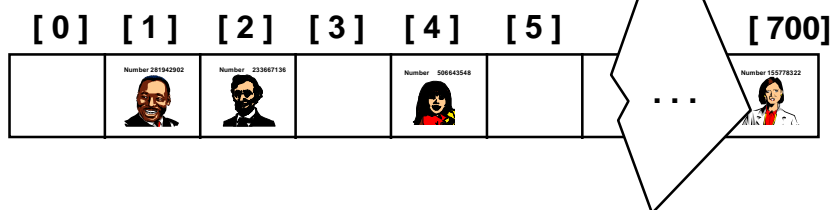
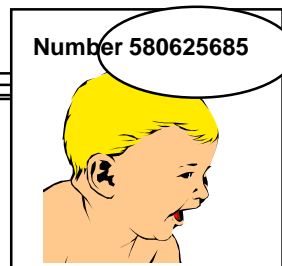
When a hash table is being used as a dictionary, some of the array locations are in use, and other spots are "empty", waiting for a new entry to come along.

Oftentimes, the empty spots are identified by a special key. For example, if all our identification numbers are positive, then we could use 0 as the Number that indicates an empty spot.

With this drawing, locations [0], [4], [6], and maybe some others would all have Number=0.

Inserting a New Record

- ❑ In order to insert a new record, the **key** must somehow be **converted to** an array **index**.
- ❑ The index is called the **hash value** of the key.



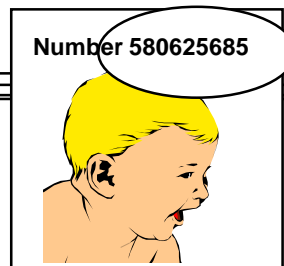
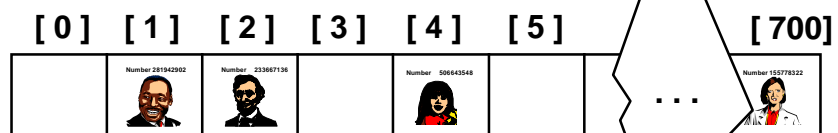
In order to insert a new entry, the key of the entry must somehow be converted to an index in the array. For our example, we must convert the key number into an index between 0 and 700. The conversion process is called hashing and the index is called the hash value of the key.

Inserting a New Record

- Typical way create a hash value:

(Number mod 701)

What is $(580625685 \text{ mod } 701)$?



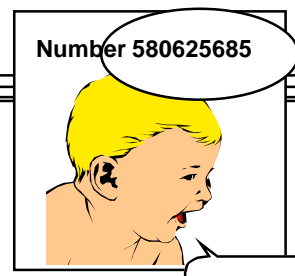
There are many ways to create hash values. Here is a typical approach.
a. Take the key mod 701 (which could be anywhere from 0 to 700).

So, quick, what is $(580,625,685 \text{ mod } 701)$?

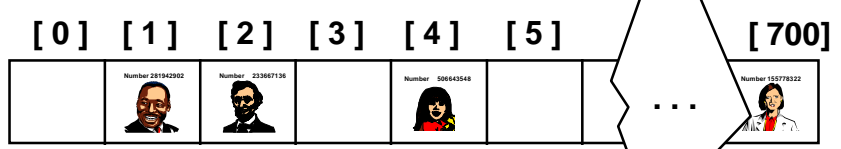
Inserting a New Record

- Typical way to create a hash value:
(Number mod 701)

What is (580625685 mod 701) ?



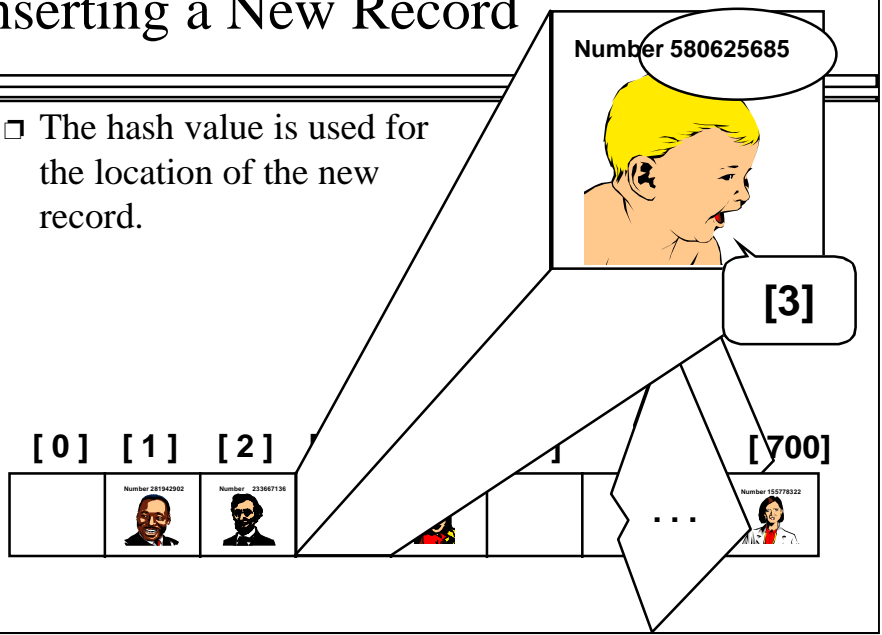
3



Three.

Inserting a New Record

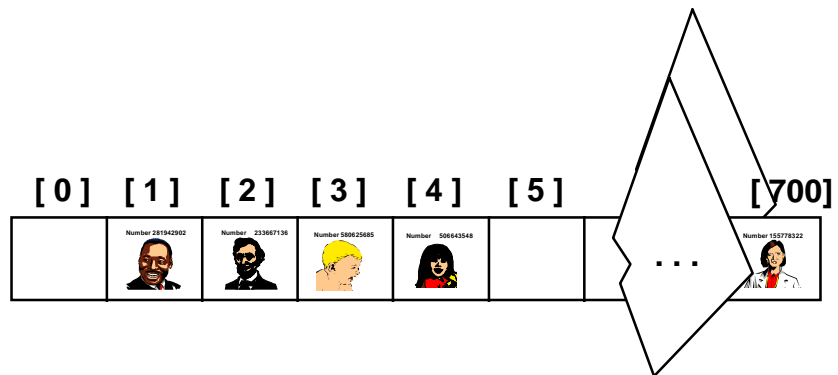
- ❑ The hash value is used for the location of the new record.



So, this new item will be placed at location [3] of the array.

Inserting a New Record

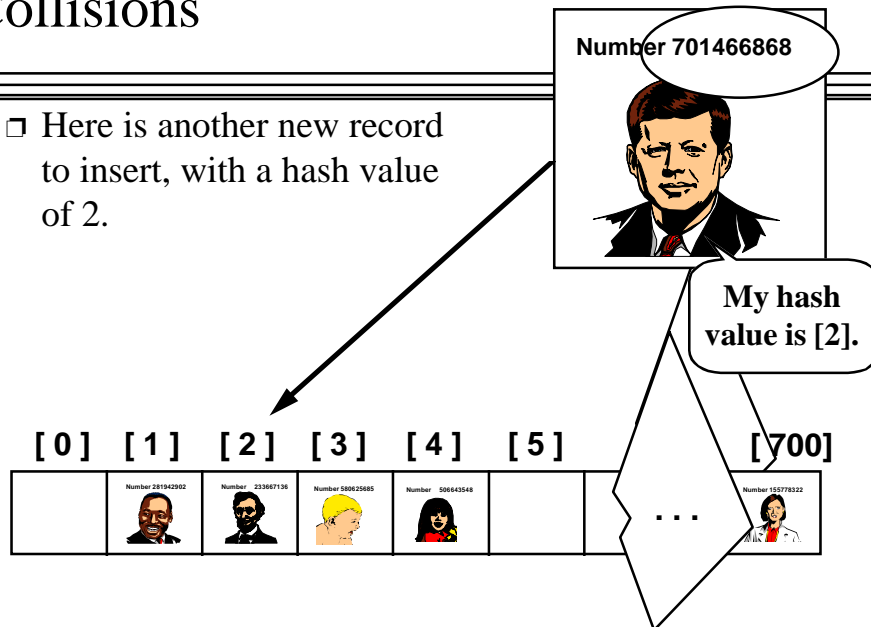
- ❑ The hash value is used for the location of the new record.



The hash value is always used to find the location for the record.

Collisions

- Here is another new record to insert, with a hash value of 2.

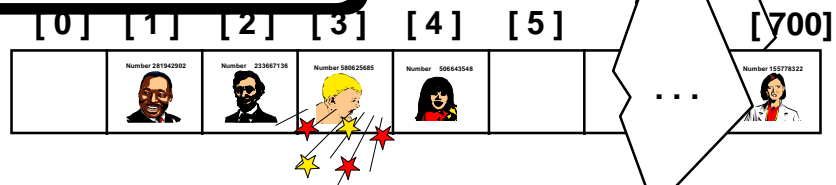


Sometimes, two different records might end up with the same hash value.

Collisions

- ❑ This is called a **collision**, because there is already another valid record at [2].

When a collision occurs, move forward until you find an empty spot.



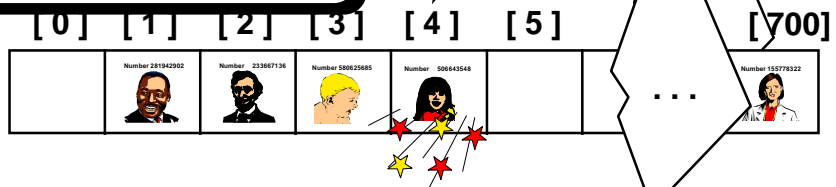
This is called a collision.

When a collision occurs, the insertion process will move forward through the array until an empty spot is found. Sometimes you will have a second collision...

Collisions

- This is called a **collision**, because there is already another valid record at [2].

When a collision occurs, move forward until you find an empty spot.

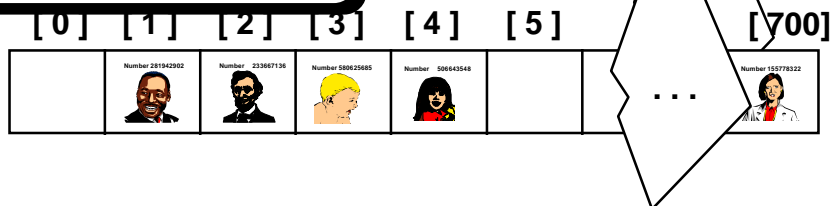


...and a third collision...

Collisions

- This is called a **collision**, because there is already another valid record at [2].

When a collision occurs, move forward until you find an empty spot.



But if there are any empty spots, eventually you will reach an empty spot, and the new item is inserted here.

Collisions

- This is called a **collision**, because there is already another valid record at [2].

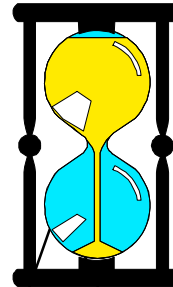
The new record goes in the empty spot.









The new record is always placed in the first available empty spot, after the hash value.

A Quiz

Where would you be placed in this table, if there is no collision? Use your social security number or some other favorite number.



[0]	[1]	[2]	[3]	[4]	[5]	[700]
	Number 281942902 	Number 233647136 	Number 508025685 	Number 50645548 	Number 701466868 	... 

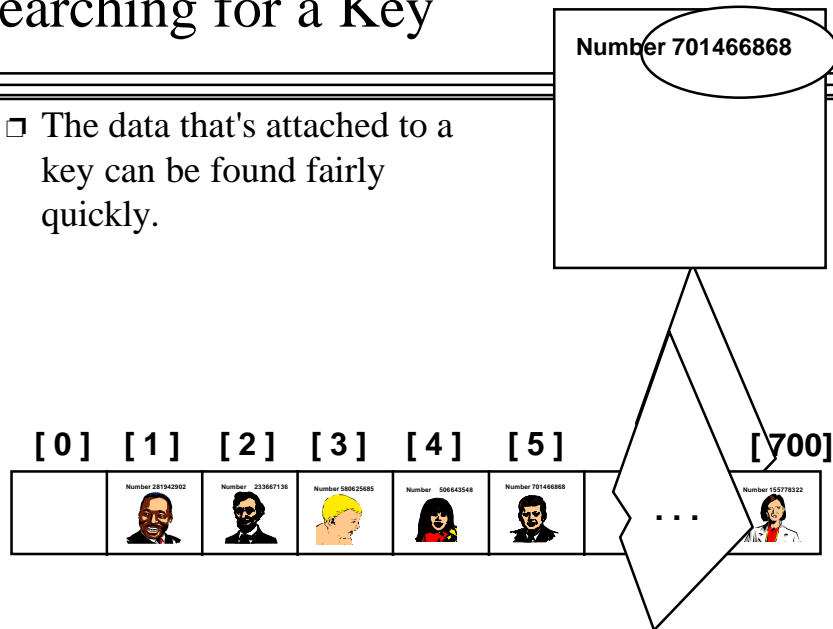
Time for another quiz . . . Did anyone end up with a hash value of 700?

Well, I did. My ID number is 155779023, which has a hash value of 700. (No, not really, but I needed to illustrate another kind of collision.)

There is a collision with the last location of the array. In this case, you would circle back to the start of the array, and try location number 0, then 1, until you find an empty spot. In this example, I would be inserted at location 0, since that location is empty.

Searching for a Key

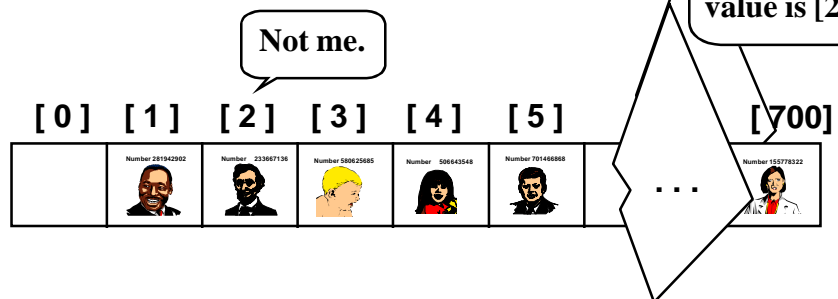
- ❑ The data that's attached to a key can be found fairly quickly.



It is fairly easy to search for a particular item based on its key.

Searching for a Key

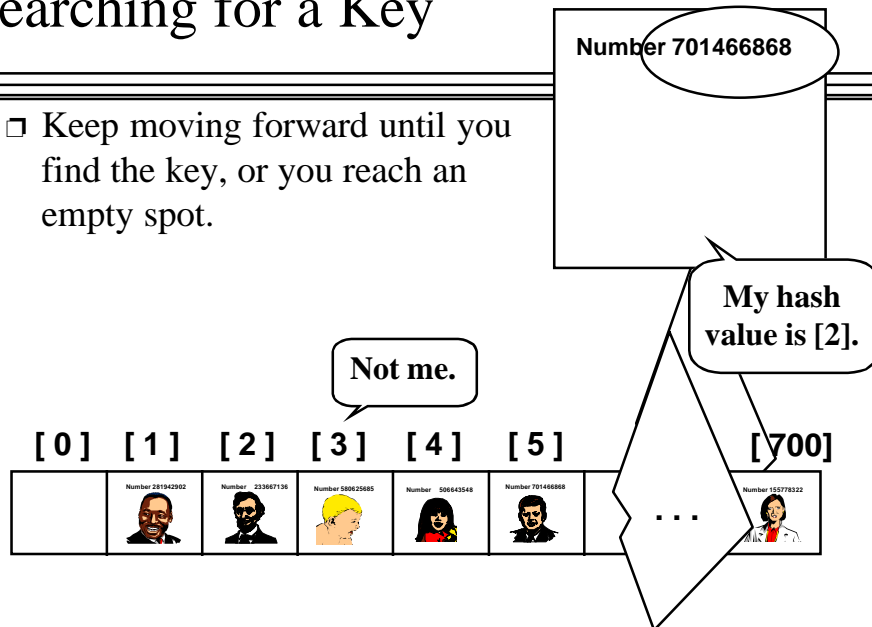
- ❑ Calculate the hash value.
- ❑ Check that location of the array for the key.



Start by computing the hash value, which is 2 in this case. Then check location 2. If location 2 has a different key than the one you are looking for, then move forward...

Searching for a Key

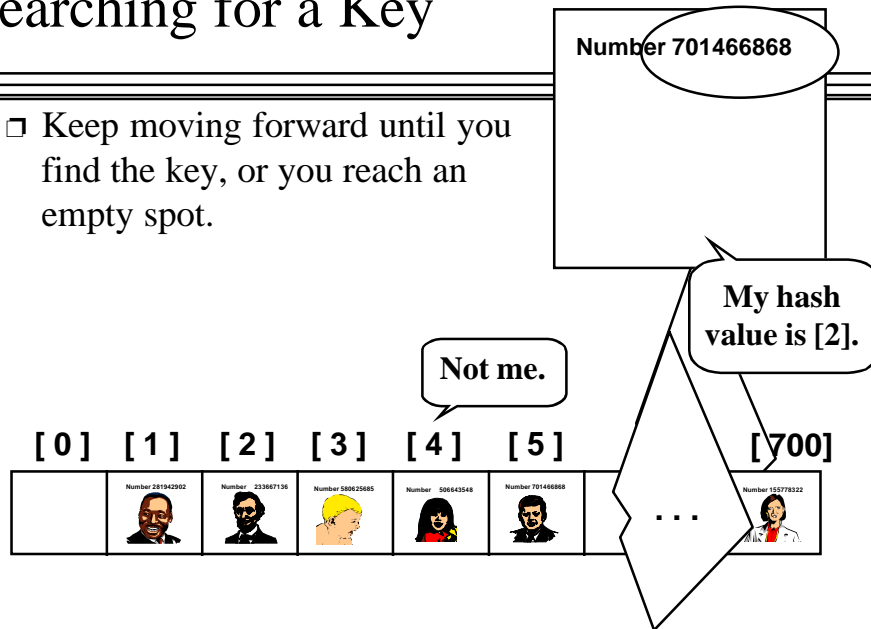
- ❑ Keep moving forward until you find the key, or you reach an empty spot.



...if the next location is not the one we are looking for, then keep moving forward...

Searching for a Key

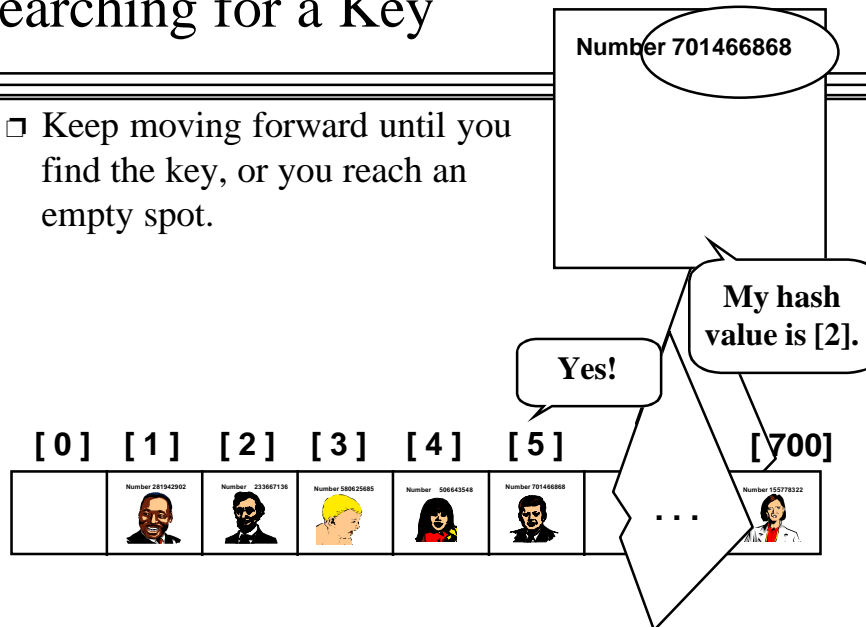
- ❑ Keep moving forward until you find the key, or you reach an empty spot.



Keep moving forward until you find the sought-after key...

Searching for a Key

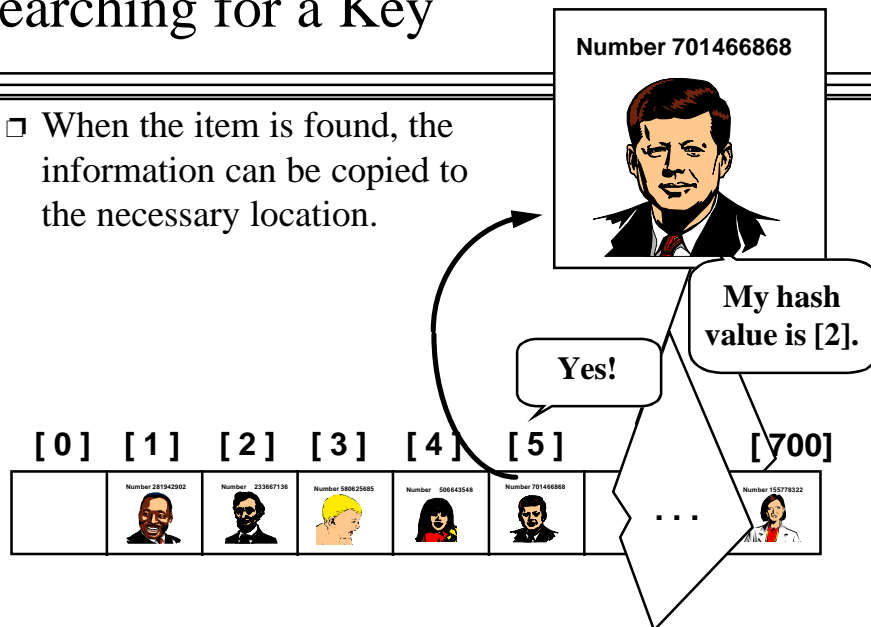
- ❑ Keep moving forward until you find the key, or you reach an empty spot.



In this case we find the key at location [5].

Searching for a Key

- When the item is found, the information can be copied to the necessary location.

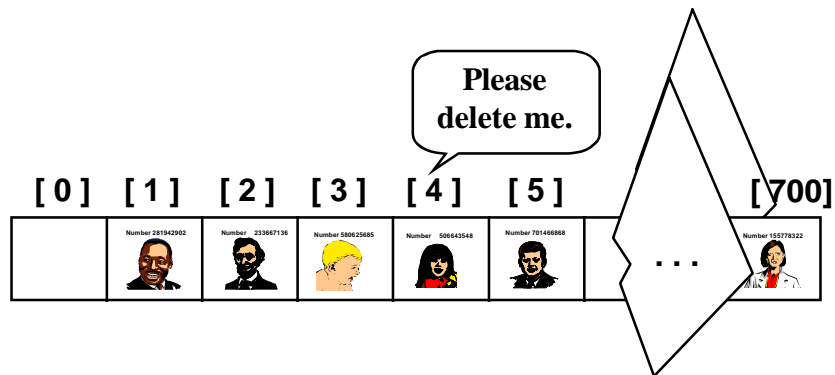


The data from location [5] can then be copied to provide the result of the search function.

What happens if a search reaches an empty spot? In that case, it can halt and indicate that the key was not in the hash table.

Deleting a Record

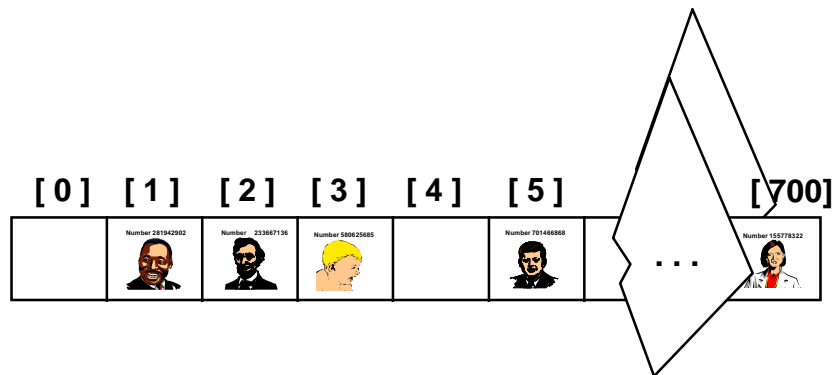
- Records may also be deleted from a hash table.



Records can be deleted from a hash table...

Deleting a Record

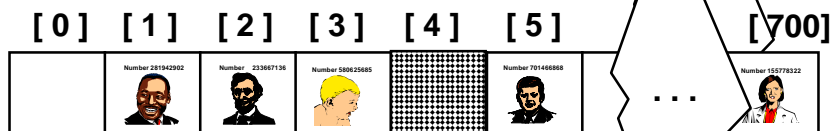
- ❑ Records may also be deleted from a hash table.
- ❑ But the location must not be left as an ordinary "empty spot" since that could interfere with searches.



But the spot of the deleted record cannot be left as an ordinary empty spot, since that would interfere with searches. (Remember that a search can stop when it reaches an empty spot.)

Deleting a Record

- ❑ Records may also be deleted from a hash table.
- ❑ But the location must not be left as an ordinary "empty spot" since that could interfere with searches.
- ❑ The location must be marked in some special way so that a search can tell that the spot used to have something in it.



Instead we must somehow mark the location as "a location that used to have something here, but no longer does."

We might do this by using some other special value for the Number field of the record.

In any case, a search can not stop when it reaches "a location that used to have something here". A search can only stop when it reaches a true empty spot.



Summary

- ❑ Hash tables store a collection of records with keys.
- ❑ The location of a record depends on the hash value of the record's key.
- ❑ When a collision occurs, the next available location is used.
- ❑ Searching for a particular key is generally quick.
- ❑ When an item is deleted, the location must be marked in a special way, so that the searches know that the spot used to be used.

A quick summary . . .

Kathy Martin
817339024



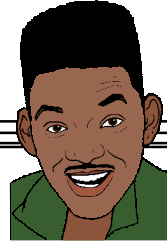
Took Data Structures in Fall 1993.
Grade A.

Hard worker. Always gets things done
on time.

Currently working for Hewlett-Packard
in Fort Collins.

To finish things off, I have hard copies of the next five slides. I tell the students that these are some records of my past students and I want to store them in a small hash table with size 5 (indexes 0 to 4). Of course, this is an unrealistic size, but it makes sure that they know the insertion, searching, and deletion algorithms. I then use five students from the front row to be the hash table locations. I insert the five items, remove Bill Clinton and do three searches (for Will Smith, Bill Clinton, and Elizabeth).

Will Smith
506643973

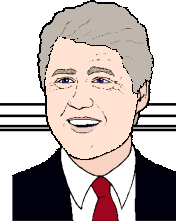


Took Data Structures in Fall 1995.
Grade A.

A bit of a goof-off, but he comes through
in a pinch.

Currently saving the world from alien
invasion.

William "Bill" Clinton
330220393



Took Data Structures in Fall 1995.
Grade B-.

Gets along with most
people well.

Currently working for federal government.

Elizabeth Windsor
092223340

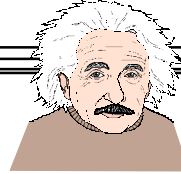


Took Data Structures in Fall 1995.
Grade B-.

Prefers to be called “Elizabeth II” or “Her Majesty.” Has some family problems.

Currently working in public relations
near London.

Al Einstein
699200102



Took CSCI 2270 in Fall 1995.
Grade F.

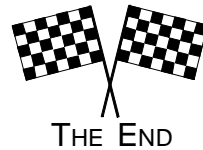
In spite of poor grade, I think there is
good academic ability in AI.

Currently a well-known advocate for
peace.

Presentation copyright 1997 Addison Wesley Longman,
For use with *Data Structures and Other Objects Using C++*
by Michael Main and Walter Savitch.

Some artwork in the presentation is used with permission from Presentation Task Force
(copyright New Vision Technologies Inc) and Corel Gallery Clipart Catalog (copyright
Corel Corporation, 3G Graphics Inc, Archive Arts, Cartesia Software, Image Club
Graphics Inc, One Mile Up Inc, TechPool Studios, Totem Graphics Inc).

Students and instructors who use *Data Structures and Other Objects Using C++* are welcome
to use this presentation however they see fit, so long as this copyright notice remains
intact.



Feel free to send your ideas to:

Michael Main

main@colorado.edu