Hash Maps

Hash Functions, Values, Associative Containers, and Maps

Contents

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Associative Containers

- Associative Containers are a type of container object that allow us to associate one piece of data to other pieces of data.
  - In a way an array is an Associative Container that associates the data in the array to its index.

- Hash maps are just one type of Associative Container.

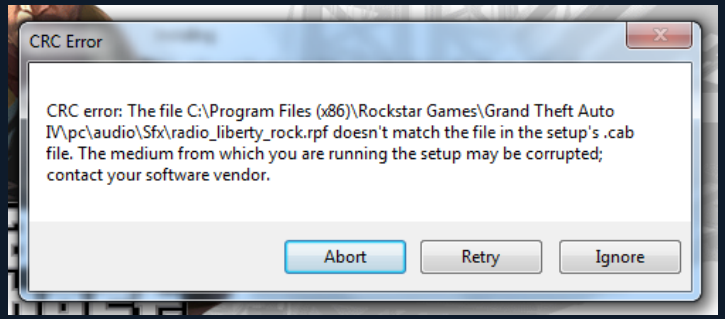
What is Hashing?

- Hashing is a technique that maps an arbitrary amount of data to a value with a fixed length.
  - Often used to map a char array to an int.
  - Example: A name like “John Morgan” might hash to a value like 1043.
- The data is passed into a “hash function” where it is processed and converted into a “hash value”.
- The same input always produces the same output.
  - Hashing “John Morgan” will always produce 1043 (unless the hash function is changed).
- Different input produces different output.
  - Example: “Alex Roberts” hashes to 1163, but “Simon Smith” hashes to 1067.

What is Hashing?

- Often used to compare large amounts of data.
  - E.g. It's faster to compare two integers than two char arrays, although hashing itself is expensive so this is only useful if you're comparing the same strings a lot.

- A similar technique is used to check for data loss in files that have been sent online or copied from a disk, this is called a Cyclic Redundancy Check (CRC)



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Hash Functions

- Takes the data in as an argument and returns the hash value.
- There are lots of different hash functions that produce different results.
  - This is a very simple example, there are much better ones out there.

```
FUNCTION HashString(string_to_hash)
    hash = 0
    FOR each character in the string_to_hash
        Add the character to hash
    END FOR
    RETURN hash
END FUNCTION
```

Hash Maps

- Hash maps (or hash tables) are a type of container that stores two associated pieces of data as a "pair".
  - For example it might store a person's name and address.
- One of the two pieces of data is commonly called the "value".
  - This is the data we might need to retrieve later, e.g. their address.
- The other is usually referred to as the "key".
  - This is used to retrieve the value, e.g. their name.
- To get the value, we hash the key and use the result as an index into an array stored inside the hash map.

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Pair Class

- A Pair class will need:
  - A constructor that takes in a value and its key and stores them both.
  - A method to return the key.
  - A method to return the value.

```
class Pair
{
public:
    Pair(char* a_key, char* a_value);
    ~Pair();

    // Returns the Pair's key.
    const char* GetKey() const;

    // Returns the Pair's Value.
    const char* GetValue() const;

private:
    Pair(const Pair &obj);

    char* m_key;
    char* m_value;
};
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Back to Hash Maps

- A hash map can store its data in an array.
- A hash map will need functions to:
  - Add a new value.
  - Get a value by its key.
- Optionally it might have ways to hash and store different types of data and methods to iterate through the map.

Adding to a Hash Map

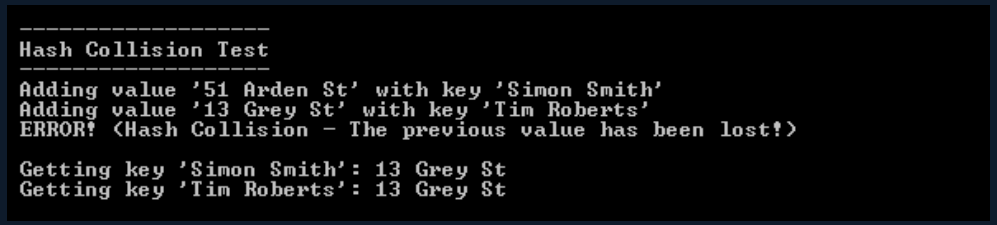
- Hash the key to get an integer.
- Use the hash to calculate an index into the map:
  - unsigned int index = (hash % MAP_SIZE);
- Create a new Pair containing the value and key and store it at the index:
  - m_map[index] = new Pair(key, value);

Getting values from a Hash Map

- Getting a value back out of the table is similar:
- Hash the key to get an integer.
- Use the hash to calculate the index in the map where the value is stored:
  - unsigned int index = (hash % MAP_SIZE);
- Return the value:
  - return m_map[index]->GetValue();

Caveat - Hash Collisions

- The big disadvantage to hash maps is that sometimes different values hash to the same key. This is called a “Collision”.



- When a hash collision occurs, both keys will map to the same index within the table and one will overwrite the other.
  - Good hashing functions will create fewer collisions.
  - One solution is to put the colliding value in the next available index, this is called “open addressing”.

Summary

- Hash maps are good for storing a large number of strings or other data with arbitrary sizes.
  - For example storing names or addresses.
  - Especially useful in games with a lot of dialogue or text.

```
--------------------
Hash Table Test
--------------------
Adding value '16 Collins St' with key 'John Morgan'
Adding value '3 Albert Rd' with key 'Alex Roberts'
Adding value '3/21 Green Ave' with key 'Margaret Adams'

Getting key 'John Morgan': 16 Collins St
Getting key 'Alex Roberts': 3 Albert Rd
Getting key 'Margaret Adams': 3/21 Green Ave
```

References

- Eric Suh . *Hash Tables at cprogramming.com*
  - http://www.cprogramming.com/tutorial/computerscienctheory/hash-table.html

- More hash functions:
  - http://www.cse.yorku.ca/~oz/hash.html

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX