Code Exchange

C++ Tutorial - Hash Tables

Posted by oneleggedredcow in Code Exchange on May 23, 2013 10:06:41 PM

Introduction

In the previous tutorial, we talked about constants. By using a constant, you are letting everyone else know that a given value will not change. Constants can be applied to variables, function parameters and functions themselves.

In this tutorial, we are going to talk about hash tables, specifically the STL unordered_map. I like to think of hash tables as a more general form of an array. Arrays are indexed by small integers, whereas hash tables can be indexed by anything.

Code

Let's start out with a quick example:

```
#include <unordered_map>
#include <string>

using namespace std;


int main()
{
    unordered_map<string, string> hashtable;
    hashtable.emplace("www.element14.com", "184.51.49.225");

    cout << "IP Address: " << hashtable["www.element14.com"] << endl;

    return 0;
}
```

In this example, we create a hash table, which is called an unordered_map in c++. The first type within the angled brackets is the key – which is analogous to the index of an array. The second type within the angled brackets is the value –

which is analogous to the value stored in an array. In this example, both the key and the value are strings. In the hash table, we are going to store the url of a website and the IP address that it resolves to. So, the next line adds `"www.element14.com"` to the hash table. The word "emplace" means "to put into place or position". Now comes the exciting part, to access the information in the hash table, we give it the key (`"www.element14.com"`) and it gives us the value (`"184.51.49.225"`). This is similar to how an array works, where we give it an index (i.e. 0) and it gives us back a value (i.e. `"184.51.49.225"`).

The unordered_map also has an insert function, but it requires slightly more code to use:

```
hashtable.insert(make_pair("www.newark.com", "184.51.50.121"));
```

This is because the insert function is expecting a key/value pair that has already been created (using make_pair), whereas the emplace function creates the key/value pair for you.

To display all of the data in the hash table, do something like this:

```
for (auto itr = hashtable.begin(); itr != hashtable.end(); itr++)
{
    cout << (*itr).first << ": " << (*itr).second << endl;
}
```

This should look familiar to the way that the STL list/vectors worked. There is also a condensed version:

```
for (auto &itr : hashtable)
{
    cout << itr.first << ": " << itr.second << endl;
}
```

Which saves some typing.

Under the Covers

Hash tables are implemented by converting the key object into a number. This number then points to a bucket where the value is stored. Once the correct bucket is found, every item in the bucket is compared with the key, to see if it is the matching item.

Therefore the performance of a hash table is very dependent on the function that transforms the key into a number (referred to as the hash function). A good hash function will randomly and uniformly distribute the given keys into buckets. So, that the comparison between a key and other objects within the same bucket is minimized.

This means that hash tables are extremely fast at inserting, removing and retrieving entries. In the average case, the time is constant, which means that they will perform very well, even with a large number of entries.

Custom Classes

The incredibly powerful part of hash tables is that anything can be a key, even custom classe


```
class Make
{
public:
    string Name;


    Make(string name)
    {
        Name = name;
    }


    bool operator==(const Make &make) const
    {
        return Name == make.Name;
    }
};


class Model
{
public:
    string Name;
    int Year;


    Model(string name, int year)
```

```
        {
            Name = name;
            Year = year;
        }


        bool operator==(const Model &model) const
        {
            return (Name == model.Name && Year == model.Year);
        }
    };


    class ModelHash
    {
    public:
        size_t operator()(const Model &model) const
        {
            return hash<string>()(model.Name) ^ hash<int>()(model.Year);
        }
    };

    int main()
    {
        unordered_map<Model, Make, ModelHash> model2make;


        Model camry2005("Camry", 2005);
        Model tercel1993("Tercel", 1993);

        Make toyota("Toyota");

        model2make.emplace(camry2005, toyota);
        model2make.emplace(tercel1993, toyota);


        for (auto &itr : model2make)
        {
            cout << itr.first.Name << " " << itr.first.Year << ": " << itr.second.Name
            << endl;
        }
```

```
    return 0;
}
```

First, we declare two classes: one to store the make of a car and one to store the model of a car. The `ModelHash` function converts the `Model` class into a number. This allows the hash table to decide which bucket to use to store the object. The `Model` class has two pieces of information: the Name of the model and the Year that it was produced. When we generate the number that will represent the hash table in the class, we want to use all of the variables that uniquely define the class. So, in this case we will use both the Name and the Year variables. If you look closely, the two variables are combined using the carrot operator (^), which performs an eXclusive OR. This is a very common trick for combining two variables in hash function.

The rest of the code look fairly similar to the first example. The only weird part, is you might be wondering why we went from model to make and not the other way around. Well, the keys for hash tables need to be unique. (This is similar to an array.) If we used the make of the car, it would not be unique. To get around this, we will store a vector of `Model`s for each `Make`:

```
class MakeHash
{
public:
    size_t operator()(const Make &make) const
    {
        return hash<string>()(make.Name);
    }
};

int main()
{
    unordered_map<Make, vector<Model>, MakeHash> make2model;


    vector<Model> toyotaModels;
    toyotaModels.push_back(camry2005);


    make2model.emplace(toyota, toyotaModels);
```

```
        make2model[toyota].push_back(tercel1993);


        for (auto &make : make2model)
        {
            cout << make.first.Name << endl;

            vector<Model> models = make.second;

            for(auto &model : models)
            {
                cout << " " << model.Name << " " << model.Year << endl;
            }
        }


        return 0;
    }
```

Now the data structure should seem more familiar where the manufacturer of the car maps to the model of cars that it produces.

Summary

In this tutorial, we learned about hash tables, which can be thought of as a more general version of an array.

This is our last tutorial. Next week there will be a recap of what we went over and a quiz to test your new skills and I even heard a rumor that there will be some prizes involved…

If you have any questions or comments about what was covered here, post them to the comments. I watch them closely and will respond and try to help you out.

Tutorial Index

69543 Views Categories: Code Exchange
Tags: c++, programming, code, raspberry_pi, rpi, raspbpi, code_exchange, learn++, learnc++

8 Comments

Login or Register to comment

creuben May 13, 2015 12:12 AM

A few errors:

```
include <string> //needed to get code to work


Make toyota("Toyota", "Japan");//doesn't compile. Two arguments given, only one accepted in constructor. Need to change to:
Make toyota("Toyota");//
```

Like (0)

creuben May 13, 2015 11:45 AM

Also, any chance you could explain a few things here:

1) Why the == after the return variable declaration and before the () parameter, and why the second mention of const after the () parameter in the operator functions, eg.

```
bool operator==(const Make &make) const
```

2) and in this one why the initial set of () before the parameter in the parentheses?

```
size_t operator()(const Model &model) const
```

Thanks for the tutorial. Edit: here's a link for more info about point 1 (above), and of course, about C++ in general:

http://en.cppreference.com/w/cpp/language/operator_comparison

Like (0)

oneleggedredcow May 17, 2015 5:32 PM (in response to creuben)

Thanks for your comments on the article!

1) The previous tutorial on constants goes over what the const keyword means in various locations:
C++ Tutorial - Constants

In this case, the first const states that the value passed in will not be changed. The second const states that no class member variable will be changed when the function is executed. (There will be no side effects.)

2) When overloading an operator, you need to specify what operator you are overloading. In this case, I want to overload the parenthesis operator, so that's the syntax to specify it. (So, think about the () like it was +, -, *, ==, etc.)

Like (0)

creuben May 20, 2015 3:39 PM (in response to oneleggedredcow)

Thanks for these responses. On 1), indeed you covered this. On 2) is the overloading happening because C++ has functions for the various operators with the function names in the format of 'operator==', 'operator()', etc. such that these functions are being overloaded within your class? So in this case you want == to check to see if two entries are equivalent and with the () one you want to check that the contents in the parentheses are exclusive or? So effectively, you are modifying the C++ default functionality of the operators?

Sorry for the rudimentary questions. Your tutorial is quite rich and concise, and I am probably going to need a slower walk through it all. Do you have a good source to recommend for further and more detailed study of C++? Thanks in advance.

Like (0)

oneleggedredcow May 21, 2015 11:45 PM (in response to creuben)

By default, various operators like == or * are not defined

for any arbitrary class that you create. So, if you created a new class called Foo:

```
class Foo
{
public:
  int Value;


  Foo(int value)
  {
  Value = value;
  }
};
```

And then try to compare it with ==, you will get a compile error:

```
  Foo a(1);
  Foo b(1);

  if (a == b) { cout << "Equal" << endl; }
```

This is because the compiler has no idea how to do that operation. I think the best example of this is with something like multiplication for a 2x2 matrix class. There are really two ways that could reasonable make sense. You could do a matrix multiplication or do an element-wise multiplication.

Matrix:

```
[ 1  2 ] * [ 1  2 ] = [  7  10 ]
[ 3  4 ]   [ 3  4 ]   [ 15  22 ]
```

Element-Wise:

```
[ 1  2 ] * [ 1  2 ] = [  1   4 ]
[ 3  4 ]   [ 3  4 ]   [  9  16 ]
```

I like to think of the operators like just another function. You could achieve the same effect by creating a function

called IsEqual or Multiply. However, sometimes it makes your code more readable if you substitute in == or * instead. That's the real power here. It makes your code more readable.

The case of the unordered_map is a lot like the case that I mentioned in the templates article:

C++ Tutorial - Templates

In the unordered_map case, I need to provide two pieces of information, the first is a hash function that returns a size_t. (size_t is really just a 64-bit integer.) This is the lookup for what bucket to find the value in and is computed by the operator() function on MakeHash. The second piece is comparing to see if the two values are equal once you are within the correct bucket. This is provided by the operator== function on Make. These operators are provided so that the unordered_map data structure knows how to function with any arbitrary class.

Yeah, the tutorials are pretty dense.

When I learned c++, I remember liking the Deitel & Deitel book the best (it was the 3rd edition back then), but I'm not sure how they've held up since then.

Hopefully that helps.

Like (0)

creuben May 22, 2015 7:51 PM

Thanks for that. I found this blog on the topic of books for learning C++:
http://stackoverflow.com/questions/388242/
Also, I'm discovering the difference between bringing Armadillo into Raspbian (Raspberry Pi) vs. into VS 2013 on Windows, 10 minutes (which was mostly just install time) vs. several hours and counting.

Any recommendations for learning that sort of stuff? Eg. how to correctly set up dependencies, libraries, .dlls, that sort of thing? Or is that wishful thinking? I encountered the same nightmare working Java in Netbeans or Eclipse a few

years back. In other words, it's one thing to learn the language, it's another to learn how to put everyone else's parts together so you don't have to build everything yourself and refine it forever until you optimize your run time. In other words, I ain't recreating Armadillo. Any points on how to learn that stuff? Or is that just the way it goes? I know a chief problem there is that unless you know just how to search, the answers you find apply only partly to your problem--and frequently partial solutions don't work--you need the whole solution, which is almost always unique for everyone's' uniquely composited problem.

That will be my last question here, I promise. It is just that it can be discouraging to learn to program only to find out you don't get far with it in the real world of releasing software or building something outside of the confines of prototyping environments. I can do a lot in some nice environment like Octave, but if I have to compile and build anything using the wonderful work of others' libraries, flip a coin as to whether I am going to get anywhere in a given environment--at least not without extensive delays. If I have a method to attack that sort of problem, however, I should think it would be well worth the effort. Again, thanks.

Like (0)

oneleggedredcow May 26, 2015 7:47 PM (in response to creuben)

That's a great question, and unfortunately one that I don't have a great answer for. I know exactly what you mean: struggling for hours to get something simple and fundamental working. I'll admit that I haven't found a great solution on how to get around that, and it seems like every scenario is different enough that the same tricks don't necessarily work. It usually just comes down to good old fashion debugging and googling. Sorry I don't have a much better answer than that.

Post as many questions as you have! No need to hold back on asking them. We've all gone through the same struggles learning a new language.

Like (0)

oleslaw Nov 9, 2015 5:47 AM

thanks for solved this problem

Like (0)

Login or **Register** to comment

Topics

3D Printing
Arduino
Business of Engineering
Embedded
Industrial Automation & Robotics
Internet of Things
Open Source Hardware
Power & Energy
Raspberry Pi
Raspberry Pi Accessories
Raspberry Pi Projects
Get Started With Pi
Sensors/MEMS
Single Board Computers
Test and Measurement
Wearable Technology

Resources

Ask Industry Experts
The Ben Heck Show
CadSoft EAGLE Design
Code Exchange
Design Challenges
Feedback & Support
Learning Center
Legislation
Manufacturers
News
PCB Prototyping
Purchasing Solutions
RoadTest
STEM Academy
Watch element14 TV
Webinars and Training

Content

Blog posts
Documents
Discussions
Polls
Videos
Events

Design Center

Products

Analog Demo Boards
Development Platforms & Kits
Emulation & Debugging

Software

IDEs & Compilers
OS & Middleware
Test & Simulation
EDA Tools

Featured Technologies

Atmel Xplained
CircuitStudio PCB Design
Freescale Freedom Platform
MATLAB for Students
LPCXPRESSO EcoSystem
Single Board Computers

Members

Member Directory
Member of the Month
Ranking & Points
Ask Industry Experts
Top Members
Why Join?
This week on element14

Store

Visit store for United States
Choose a different store

element14 is the first online community specifically for engineers. Connect with your peers and get expert answers to your questions.

About Us
Our Evolution

Follow Us