CS3451: C++ Tutorial

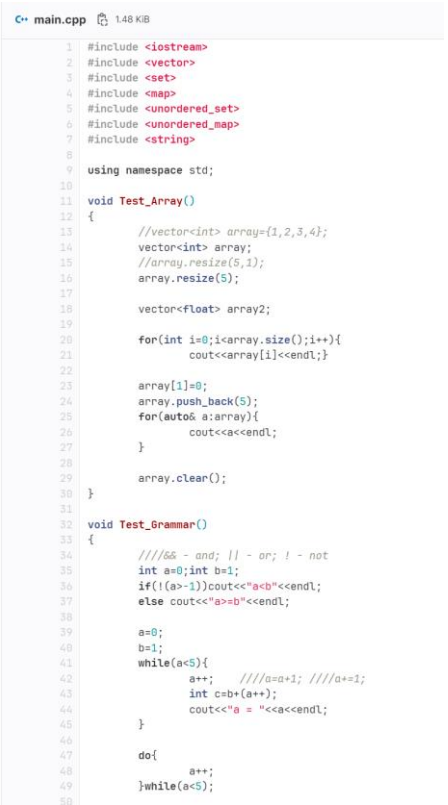
Bo Zhu

School of Interactive Computing

Georgia Institute of Technology

C++ Tutorial Code

https://gitlab.com/boolzhu/dartmouth-cg-starter-code/-/blob/master/tutorials/tutorial_cpp101/main.cpp


main.cpp 1.48 KiB

```cpp
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <unordered_set>
#include <unordered_map>
#include <string>

using namespace std;

void Test_Array()
{
	//vector<int> array={1,2,3,4};
	vector<int> array;
	//array.resize(5,1);
	array.resize(5);

	vector<float> array2;

	for(int i=0;i<array.size();i++){
		cout<<array[i]<<endl;}

	array[1]=0;
	array.push_back(5);
	for(auto& a:array){
		cout<<a<<endl;
	}

	array.clear();
}

void Test_Grammar()
{
	////&& - and; || - or; ! - not
	int a=0;int b=1;
	if(!(a>-1))cout<<"a<b"<<endl;
	else cout<<"a>=b"<<endl;

	a=0;
	b=1;
	while(a<5){
		a++;	////a=a+1; ////a+=1;
		int c=b+(a++);
		cout<<"a = "<<a<<endl;
	}

	do{
		a++;
	}while(a<5);
```

Variables

```
int a=0;int b=1;

//vector<int> array={1,2,3,4};
vector<int> array;

vector<float> array2;
```

Data Types:

bool, char, int, float, double…

auto a = 5;

(need initialization)

~~auto a;~~

Operators

+, - *, /, %

==, !=, >, <, >=, <=

&&, ||, !

+=, -=, *=, /=, %=

++, - -

Operators

```
int a=0;
int b=1;
if ( !(a>-1) ) {
        cout << "a<=-1" << endl;
} else {
        cout << "a>-1" << endl;
}

if ((a>=0) && (b>=0)) {
        cout << "Both a & b are non-negative." << endl;
}
if ((a==0) || (b>=0)) {
        cout << "Either a or b is zero." << endl;
}
```

Operators

Note:

++a returns the value of a after it has been incremented.

a++ returns the value of a before incrementing.

Operators

```
int x = 3;
int y, z;
y = x++; //returns the value before incrementing.
z = ++x; //returns the value after it has been incremented.
cout << x << " " << y << " " << z << endl;
```

Loops

```
for (int i = 0; i < 10; i++) {
…
}
```

```
for (auto& a : array) {
… (for changing a)
}
```

```
while (a < 5) {
        a++; …
}
```

```
do {
        a++; …
} while (a<5);
```

Loops

```
a=0;
b=1;
while(a<3){
        a++;
        int c=b+(a++);
        cout << "c = " << c << ", ";
        cout << "a = " << a << endl;
}
```

Conditionals

```
int d=0;
// int d=1;
// int d=2;
switch(d){
case 0:
        cout<<"case 0"<<endl;
        break;
case 1:
        cout<<"case 1"<<endl;
        break;
default:
        cout<<"case default"<<endl;
        break;
}
```

Arrays

					0			
					0	2	2	
	0				0	2	2	
	0				0	2	2	5

```
//vector<int> array={1,2,3,4};
vector<int> array;
vector<float> array2;
array.resize(5,1);
array.resize(5);
array.resize(6);
array.resize(8,2)

for(int i=0;i<array.size();i++){
        cout << array[i] << " ";
}
cout << endl;

array[1]=0;
array.push_back(5);
for(auto& a:array){
        cout << a << " ";
}
cout << endl;

array.clear();
```

Containers

set: https://cplusplus.com/reference/set/set/

map: https://cplusplus.com/reference/map/map/

unordered_set:
https://cplusplus.com/reference/unordered_set/unordered_set/

unordered_map:
https://cplusplus.com/reference/unordered_map/unordered_map/

Set vs. Unordered_set

Differences :

	set	unordered_set
Ordering	increasing order (by default)	no ordering
Implementation	Self balancing BST like Red-Black Tree	Hash Table
search time	log(n)	O(1) -> Average O(n) -> Worst Case
Insertion time	log(n) + Rebalance	Same as search
Deletion time	log(n) + Rebalance	Same as search

Set

```
// SET
set<int> s={1,3,5,7};
s.insert(2);
s.insert(5);
//s.erase(2);
//s.clear();

cout << "set elements: ";
for(auto& a:s){
        cout<< a << " ";
}
cout << endl;

auto result=s.find(6);
if(result!=s.end()){
        cout<<"find "<<(*result)<<endl;
}
```

Unordered_set

```
// UNORDERED_SET
unordered_set<int> hashset;
hashset = {1,3,5,7};
hashset.insert(2);
hashset.insert(5);
//hashset.erase(3);
cout << "hashset elements: ";
for(auto& a: hashset){
        cout << a << " ";
}
cout << endl;

auto result3=s.find(6);
if(result!=s.end()){
        cout<<"find "<<(*result3)<<endl;
}
```

Map

```
// MAP
cout << "map elements: " << endl;
map<int,string> m;
m[1]="one";
m[3]="three";
m[2]="two";
for(auto& iter:m){
        std::cout<<"map ele key: "<<iter.first
                <<", value: "<<iter.second<<endl;
}

auto result2=m.find(1);
if(result2!=m.end()){
        cout<<(*result2).first<<", "<<(*result2).second<<endl;
}
```

Unordered_map

```
// UNORDERED_MAP
cout << "hashmap elements: " << endl;
unordered_map<int,string> hashmap;
hashmap[1]="one";
hashmap[3]="three";
hashmap[2]="two";
for(auto& iter:hashmap){
        std::cout<<"hashmap ele key: "<<iter.first
                <<", value: "<<iter.second<<endl;
}
auto result4=m.find(1);
if(result4!=m.end()){
        cout<<(*result4).first<<", "<<(*result4).second<<endl;
}
```

Namespace

```
using namespace std;

cout << “helloworld”;


std::cout << “helloworld”;

Eigen::MatrixXd m(2,5);
```

Matrix Tutorial

https://gitlab.com/boolzhu/dartmouth-cg-starter-code/-/blob/master/tutorials/tutorial_matrix/main.cpp

main.cpp 3.21 KiB

```cpp
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <unordered_set>
#include <unordered_map>
#include <string>
#include <Eigen/Dense>

using namespace std;

void Test_Pointer_Reference()
{
        ////reference and pointer for variable
        int a=1;
        int b=a;
        int& c=a;       ////reference
        int* d=&a;      ////pointer

        std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

        a=2;
        std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

        b=3;
        std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

        c=4;
        std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

        *d=5;
        std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

        ////reference and pointer for array
        std::vector<int> array_a={1,1,1};
        std::vector<int> array_b=array_a;
        std::vector<int>& array_c=array_a;
        std::vector<int>* array_d=&array_a;
```

Pointers

- A variable whose value is the address of another variable
- Declaring a pointer: int* p, int *p, float* p, float *p
- Storing the address of another variable: p = &var
- Accessing the value of the variable being pointed to: cout << *p << endl;

References

An alias for an already existing variable

int i = 1;

int& ref = i;

Often used in functions to avoid duplicating variables

```
void Get_Triangle_Mesh_Edges(const TriangleMesh<3>& triangle_mesh,std::unordered_set<Vector2i>& hashset)
{

void Create_Cube_Mesh(double length,TriangleMesh<3>& tri_mesh)
{
```

References

- Call by value:

```
void Create_Cube_Mesh(double length,TriangleMesh<3>& tri_mesh)
{
```

- Call by reference:

```
void Get_Triangle_Mesh_Edges(const TriangleMesh<3>& triangle_mesh,std::unordered_set<Vector2i>& hashset)
{
```

- Call by pointer

```
void MyFunc(int *x) { *x = 1; }
```

Pointers vs References

- You can have null pointers but not null references (nullptr)
- You can have a pointer to another pointer but not a reference to another reference
- References cannot be reinitialized

```
////reference and pointer for variable
int a=1;
int b=a;
int& c=a;       ////reference
int* d=&a;      ////pointer

std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

a=2;
std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

b=3;
std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

c=4;
std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;

*d=5;
std::cout<<"a = "<<a<<", b = "<<b<<", c = "<<c<<", d = "<<*d<<std::endl;
```

Pointers vs References

- You can have null pointers but not null references
- You can have a pointer to another pointer but not a reference to another reference
- References cannot be reinitialized
- Reference and pointer for arrays and array elements are analogous

```
////reference and pointer for array
std::vector<int> array_a={1,1,1};
std::vector<int> array_b=array_a;
std::vector<int>& array_c=array_a;
std::vector<int>* array_d=&array_a;

std::cout<<"\n\narray_a: ";for(auto& x:array_a)std::cout<<x<<", ";
std::cout<<"\narray_b: ";for(auto& x:array_b)std::cout<<x<<", ";
std::cout<<"\narray_c: ";for(auto& x:array_c)std::cout<<x<<", ";
std::cout<<"\narray_d: ";for(auto& x:(*array_d))std::cout<<x<<", ";

array_a[0]=2;
std::cout<<"\n\narray_a: ";for(auto& x:array_a)std::cout<<x<<", ";
std::cout<<"\narray_b: ";for(auto& x:array_b)std::cout<<x<<", ";
std::cout<<"\narray_c: ";for(auto& x:array_c)std::cout<<x<<", ";
std::cout<<"\narray_d: ";for(auto& x:(*array_d))std::cout<<x<<", ";

array_b[0]=3;
std::cout<<"\n\narray_a: ";for(auto& x:array_a)std::cout<<x<<", ";
std::cout<<"\narray_b: ";for(auto& x:array_b)std::cout<<x<<", ";
std::cout<<"\narray_c: ";for(auto& x:array_c)std::cout<<x<<", ";
std::cout<<"\narray_d: ";for(auto& x:(*array_d))std::cout<<x<<", ";

array_c[0]=4;
std::cout<<"\n\narray_a: ";for(auto& x:array_a)std::cout<<x<<", ";
std::cout<<"\narray_b: ";for(auto& x:array_b)std::cout<<x<<", ";
std::cout<<"\narray_c: ";for(auto& x:array_c)std::cout<<x<<", ";
std::cout<<"\narray_d: ";for(auto& x:(*array_d))std::cout<<x<<", ";
```

Macro

A label defined in the source code that is replaced by its value by the preprocessor before compilation

#define identifier value

// create symbolic constants

#undef identifier

```
void Test_Macro()
{
#define a 10
        std::cout<<"a="<<a<<std::endl;

#define for_loop(i,start,end) \
        for(int i=start;i<end;i++)

#define for_loop_2d(i,j,start,end) \
        for(int i=start;i<end;i++)      \
                for(int j=start;j<end;j++)

        for_loop(i,0,10){
                std::cout<<i<<", ";
        }

        for_loop_2d(i,j,0,10){
                std::cout<<"["<<i<<","<<j<<"], ";
        }
}
```

Eigen

https://eigen.tuxfamily.org/dox/modules.html

Normalize: modifies the original vector

```
void Test_Eigen()
{
        using Vector2=Eigen::Matrix<float,2,1>;
        Vector2 v=Vector2(1,2);
        Vector2 v2=Vector2(2,3);
        Vector2 v3=v+v2;
        float dot_prod=v.dot(v2);
        float norm=v.norm();
        Vector2 normal=v.normalized();
        v.normalize();

        std::cout<<"v: "<<v.transpose()<<std::endl;
}
```

Questions?