myString="PythonForBeginners"

x=myString[-1]

print(x)

s

myString="PythonForBeginners"

x=myString[0]

print(x)

p

myString="PythonForBeginners"

x=myString[0:6]

print(x)

Python

myString="PythonForBeginners"

x=myString[-5:-1]

print(x)

nner

string\_name[start\_index,end\_index,difference]

myString="PythonForBeginners"

x=myString[0:10:2]

print(x)

x=myString[0:10:3]

print(x)

PtoFr

PhFB

// map::insert (C++98)

#include <iostream>

#include <map>

int main ()

{

std::map<char,int> mymap;

// first insert function version (single parameter):

mymap.insert ( std::pair<char,int>('a',100) );

mymap.insert ( std::pair<char,int>('z',200) );

std::pair<std::map<char,int>::iterator,bool> ret;

ret = mymap.insert ( std::pair<char,int>('z',500) );

if (ret.second==false) {

std::cout << "element 'z' already existed";

std::cout << " with a value of " << ret.first->second << '\n';

}

// second insert function version (with hint position):

std::map<char,int>::iterator it = mymap.begin();

mymap.insert (it, std::pair<char,int>('b',300)); // max efficiency inserting

mymap.insert (it, std::pair<char,int>('c',400)); // no max efficiency inserting

// third insert function version (range insertion):

std::map<char,int> anothermap;

anothermap.insert(mymap.begin(),mymap.find('c'));

// showing contents:

std::cout << "mymap contains:\n";

for (it=mymap.begin(); it!=mymap.end(); ++it)

std::cout << it->first << " => " << it->second << '\n';

std::cout << "anothermap contains:\n";

for (it=anothermap.begin(); it!=anothermap.end(); ++it)

std::cout << it->first << " => " << it->second << '\n';

return 0;

}

element 'z' already existed with a value of 200

mymap contains:

a => 100

b => 300

c => 400

z => 200

anothermap contains:

a => 100

b => 300

Mymap[key] = value;

Mymap.clear();

Mymap.swap(myothermap);

Mymap.begin() mymap.rebgin() mymap.end() mymap.rend()

Mymap.size();

// map::find

#include <iostream>

#include <map>

int main ()

{

std::map<char,int> mymap;

std::map<char,int>::iterator it;

mymap['a']=50;

mymap['b']=100;

mymap['c']=150;

mymap['d']=200;

it = mymap.find('b');

if (it != mymap.end())

mymap.erase (it);

// print content:

std::cout << "elements in mymap:" << '\n';

std::cout << "a => " << mymap.find('a')->second << '\n';

std::cout << "c => " << mymap.find('c')->second << '\n';

std::cout << "d => " << mymap.find('d')->second << '\n';

return 0;

}

elements in mymap:

a => 50

c => 150

d => 200

this works easier

if (mymap.count(c)>0)

std::cout << " is an element of mymap.\n";

else

std::cout << " is not an element of mymap.\n";

#include <map>

#include <iostream>

#include <cassert>

int main(int argc, char \*\*argv)

{

std::map<std::string, int> m;

m["hello"] = 23;

// check if key is present

if (m.find("world") != m.end())

std::cout << "map contains key world!\n";

// retrieve

std::cout << m["hello"] << '\n';

std::map<std::string, int>::iterator i = m.find("hello");

assert(i != m.end());

std::cout << "Key: " << i->first << " Value: " << i->second << '\n';

return 0;

}

Output:

23

Key: hello Value: 23

sets

std::set<int> myset;

// set some initial values:

for (int i=1; i<=5; ++i) myset.insert(i\*10);

stacks/queues

std::stack<int> mystack;

mystack.push(i);

mystack.pop();

mystack.size()

.top() – for stack, .front/back() – for queue

.