Cheatography

C++ Basics Cheat Sheet

by Dockson via cheatography.com/24842/cs/6284/

Iterator for loop

```
for(std::vector<int>::iterator it = v.begin(); it != v.end(); ++it) {
    it->doSomething();
 }
```

Range-based for loop

```
vector<int> v;
for(int i : v) { // i = förväntad datatyp
  //doSomething
}
```

Lambda

```
[captures] (parameters) mutable -> return-type {body}
[] = argument att fånga (behandlas som globala variabler).
() = normala funktionsargument.
-> return-type = datatyp att returnera.
mutable = lägg till om du vill ändra på något. Non-const function.
{} = function goes here.
Lambda går att skriva helt utan (), -> return-type och mutable.
```

Hitta saker med egen jämförelse

```
find_if (InputIt first, InputIt last, UnaryPredicate boolFunction) //returns iterator to element if found, otherwise returns last
find_if_not() finns också.
```

Felhantering

```
try {
    foo() //foo kastar ett error med throw(“Feltext”) någonstans i koden
}
catch(exception& e)
{
    std::cout << e.what() << std::endl; // blir “Feltext”
}
```

Kommandoradsargument med validering

```
int main(int argc, char* argv[ ]) { // plocka ut argument.
    vector<string> arg(argv, argv + argc);
    int num_of_iterations;
    istringstream arg(arg.at(1));
    if(!(arg >> num_of_iterations)) //om arg inte kan göras till int så kastas logic_error {
        throw logic_error("Error meddelande");
    }
}
```

for_each loop med lambda

```
vector<int> v;
int sum{0};
for_each(v.begin(), v.end(), [&sum](int x) -> void {
    sum += x;
});
```

Huvudklass till subklass (Employee ← Boss)

```
Boss b = dynamic_cast<Boss>(employee);
if (b != nullptr) {
    std::cout << b -> get_bonus
}
```

Läsa från fil med ifstream

```
ifstream file(“/some/path”)
string line;
vector<string> file_content;
while (getline(file, line)) {
    file_content.push_back(line);
}
```

Konstruktorer/tilldelning

```
Ptr(Ptr const& rhs); //kopieringskostruktor
Ptr& operator = (Ptr const& rhs);
//kopieringstilldelning
Ptr(Ptr && this) //flyttkonstruktor
Ptr& operator = (Ptr && rhs(?)) //flytt tilldelning
```

Slumpa tal

```
#include <random>
std::random_device rnd;
std::uniform_int_distribution<int> num(1,100); // tal mellan 1-100
std::cout << num(rnd) << std::endl;
```