# **My C++**

# **Cheat Sheat**

Clear the input buffer

cin.clear();

Initialization

int a ; // no initialize r

double d = 7; // initialize r using the = syntax

vector<int> vi(10); // initialize r using the ( ) syntax

vector<int> vi2 {1,2,3,4}; // initialize r using the { } syntax

Type converion

int x3 = static\_cast<int>(x);

Bracket initization

signed char c[1] { -128 };

Type casting

int x1 = narrow\_cast<int>(2.9);

if(x!=int(x)) error("Not an int.");

Checking vector indexes for error

in case you want it to throw an exception, use std::vector::at1 instead of operator[]

try {

agrid.at(-1) = 5;

}

catch (const std::out\_of\_range& e) {

cout << "Out of Range error.";

}

Iterate through all array elements

for(int i : vi)

cout << "i = " << i << endl;

(member) initializer list.

Da te::Da te(int yy, int mm, int dd) // constructor

:y{yy}, m{mm}, d{dd} // note : membe r initialize rs

{

}

member function bodies in the class declaration

The obvious rule of thumb is: Don’t put member function bodies in the class declaration unless you

know that you need the performance boost from inlining tiny functions. Large functions, say five or

more lines of code, don’t benefit from inlining and make a class declaration harder to read. We rarely

inline a function that consists of more than one or two expressions.

cla ss Da te {

public:

void a dd\_da y(int n)

{

// . . .

}

Clearing incorrect cin input

else if (!cin.eof()) {

cin.clear();

cout << "invalid character ('" << char(cin.get()) << "') ignored - try again\n";

}