Cheatography

C++ Cheatsheet Cheat Sheet

by Technecure via cheatography.com/84247/cs/19902/

Common Data Types

bool	1 byte
char	1 byte
int	4 bytes (at least 2 bytes)
long int	4 bytes
long long int	8 bytes

Header Files & Common Includes

#include <filename>

#include <iostream> // cin & cout

#include <fsream> // file streams

#include <vector> // vectors

#include <string> // strings

Operators

a + b	Addition
a - b	Subtraction
a * b	Multiplication
a / b	Division
a % b	Modulus
a -= b	(a-b) store in a
a += b	(a+b) store in a
a /= b	(a/b) store in a
a *= b	(a*b) store in a
a++	(a+1) store in a
a--	(a-1) store in a

File IO

```
int main() {
    // this makes a new file stream
    fstream fileStream;
    // open text.txt to write to
    fileStream.open("test.txt", ios::out);
    if (fileStream.is_open()) {
        cout << "File opened!" << endl;
    }

    // write a line to the file
    fileStream << "Hello!\n";
    // must close to free the resource
    fileStream.close();
    return 0;
}
```

Note that you must *close* the file before you can *open* a new new one. **ios::out** means you want to write to the file and **ios::in** means you want to read from the file. You write to a *file stream* the same way you write to *cout*.

Classes

```
class some_name {
private:
     int m_some_data1;
     double m_some_data2;
public:
     // this is a constructor
     some_name(int a, double b) {
          m_some_data1 = a;
          m_some_data2 = b;
     }
     // getters
     int get_some_data1() {
          return m_some_data1;
     }
     double get_some_data2() {
          return m_some_data2;
     }
};
int main() {
     /* makes a new object called name
        which is of some_name type */
     some_name name(0, 2.1);
     return 0;
}
```

Classes are just like user defined types like **int** or **double**. When an object is created it calls the *constructor*. The constructor is a function with the same name as the class.

Comparison Operators

a < b	True if a is less than b
a <= b	True if a is less than or = to b
a > b	True if a is greater than b
a >= b	True if a is greater than or = to b
a == b	True if a equals b
a && b	True if a and b are true
a || b	True if a or b are true

Note: If they do not meet the criteria to be **true**, they are **false**

Pointers

```
int main() {
     int x = 3;
     // & gets the memory address of x
     int* pointer_to_x = &x;
     / pointers must be dereferenced with *
        before they are accessed. */
     *pointer = 5;
     return 0;
}
```

Note that pointers only hold a *memory address*. They cannot store anything else. In order to actually get the data at the address they must *dereference* it using the * operator.

Pointers and References

int* ptr = mem_address;	pointer definition
int& ref = other_var;	lvalue reference

Note: pointers hold a single memory address that you can change while a reference holds a single unchangeable memory address.

Pointers

```
int main() {
     int x = 3;
     // & gets the memory address of x
     int* pointer_to_x = &x;
     / pointers must be dereferenced with *
        before they are accessed. */
     *pointer = 5;
     return 0;
}
```

Note that pointers only hold a *memory address*. They cannot store anything else. In order to actually get the data at the address they must *dereference* it using the * operator.

Functions & Prototypes

```
void foo(); // prototype
void bar(int i); // prototype w/ params
void foo() { // foo definition
     std::cout << "Foo function\n";
}
void bar(int i) { // bar definition
     std::cout << "Bar: " << i << "\n";
}
int main() { // main definition
     foo(); // calls foo function
     bar(2); // calls bar with 2
     return 0;
}
```

All programs must have a main function. This is the first function that gets called. All functions except main() should have a prototype.