

Terminology

Cin	is the same as input
Cout	is same as print
<< and >>	point to where the information is going to.
difference between the ++ being before the variable or after	++x adds 1 to the x variable before doing the math statement, x++ adds 1 after the math statement.

setprecision(3) cout << setprecision(3) << 1.23456<< endl; 3 total numbers, 2 after the decimal =1.23

setprecision command with fixed notation cout << fixed << setprecision(3) << 1.23456 << endl; 4 total numbers, 3 after the decimal = 1.235

Math Libraries ceiling goes up to next int and floor goes to the int below
 ceil(3.5) = 4 floor(3.5)=3 Abs(-3.5) = 3.5

setw sets the field width to be used on the display

function a way to modularize code and make it reusable. They help us manage, reuse, and organize our code. must include return type, name, parameters and {}.
 int AddFive(int a) {return x+5}}

Terminology (cont)

parameter passing by value aka pass by copy, modifies a copy of the variable so that original stays the same. int addfive(int a) { return a = a +5}; int b = 5; addfive(b); shows 5

parameter passing by reference modifies the actual variable being passed in. void addfive(int& a) { a= a+5;} int b = 5; addfive(b); shows 10.

.h/.cpp

.h aka header file allows us to link our program to library code #include "main.h"

.cpp source code files - methods for the definitions listed in .h file are here

classes Student class {} within .h file, have member variable and member functions. default visibility of private

class scope private: only available within that function, public: available throughout the code and protected

.h/.cpp (cont)

Constructors a member function of a class that is executed whenever we create new objects of that class, it is named after the class and gives our member variables values.
 student();

default constructor creates an empty object, numeric values to 0, character and string values to blank "", and sets objects to empty. can be written in line or initialized list. Student() : id_(0), name_(""), email_(""){}

General Constructor creates an object with user chosen values given to the member variables. can be in line or a list but not both. Student(int id, std::string name, std::string email) : id_(id), name_(name), email_(email) {}

Setters gets details from constructors either general or default and add them to the variables list. void SetId(int id);

Getters a way for the user of the class to see the values without having direct access to them. int GetId() const {return id_;}



By [Jumpingjaden](https://cheatography.com/jumpingjaden/)

cheatography.com/jumpingjaden/

Published 26th January, 2025.

Last updated 26th January, 2025.

Page 1 of 3.

Sponsored by [Readable.com](https://readable.com)

Measure your website readability!

<https://readable.com>

.h/.cpp (cont)

structs similar to class but can group several related variables in one place.

Loops

If statement executes when the condition is true. double up on ifs and code executes for all true statements. `if(force < 10){}`

else if executes the first true statement. then skips the rest. `else if(force < 25){}`

for loops (start, end, change) `for(int i = 2; i < 12, i++) {}` `i--` for counting down

while loops iterates while a condition is true. `while (num <= 10){}`

do/while loops same as while but if the condition is false to begin with it will go through at least once. It can also cut down on code. `do() while();`

try/catch try to make this block of code work, if it doesn't use the catch statement to show an error message

throw creating an error message within a definition or within main. Any throw will work as long as catch has (...) next to it.

Overloaded Operators: classes

Overloaded Function same name different parameters, need these to be able to do basic math/comparisons on objects `student1` vs `int 5`

Member Functions - binary `Box operator+(Box const &lhs) const` adding a box variable to a rhs

Member Functions - binary - definition `Box Box::operator+(Box const &rhs) const {}.`

Member functions unary only the implicit argument is required.

Friend functions not members of the class. given access to the private member variables. used specifically when you want to be able to use an object on the rhs rather than the lhs. `friend Box operator+(Box const &lhs, Box const &rhs);`

friend function as insertion operator `friend std::ostream& operator<<(std::ostream& out, Item &item);` DEFINITION: `ostream& operator<<(ostream &out, Item &item) { out << "Name: " << item.name_ << "Description: " << item.description_ << " Price: " << item.price_; }`

scope

global can be accessed from anywhere in the program, not placed in a function

local also known as block scope. between []

function between {} of a function

namespace defined in the namespace.

overloaded << operator

```

Member Functions - Unary
Syntax for an overloaded unary operator
Only the implicit argument is required here
Box operator-()
Box Box::operator-()
return Box(-length_, -width_, -height_);
bool Item::operator==(const Item &match) const {
return (id_ == match.id_ && name_ == match.name_ && description_ == match.description_ && price_ == match.price_);
}
    
```

writing a function

- Write a function above main called `PrintEqualityResult` that accepts two `Distance` objects (variables created using the class name in `main.cpp`, it can be named anything I used `dist1` and `dist2` in void to make it easier to see the relations), as parameters. The return type should be void. The function should check to see if the two given parameters are equal using the overloaded `==` operator. If they are, print "They are equal!". If not, print "They are NOT equal!".
- Call the new function under Part 1 using the two `Distance` objects as parameters to check if it works correctly. Take a screenshot of your output and paste it below.

```

void PrintEqualityResult(Distance& dist1, Distance& dist2) {
if (dist1 == dist2)
std::cout << "They are equal!" << std::endl;
else
std::cout << "They are NOT equal!" << std::endl;
}

int main()
Distance dist1(1, 5);
Distance dist2(1, 13);
PrintEqualityResult(dist1, dist2);
}

Ex 2
Distance operator+(Distance const lhs, int inches)
{
inches += lhs.inches_;
return Distance(lhs.get_x(), inches);
}

inches += lhs.inches_; // Add the inches from the lhs (left-hand side)
Distance object to the given 'inches' value
return Distance(lhs.get_x(), inches); // Return a new Distance object
with the same feet from lhs and the modified inches
    
```



By Jumpingjaden

cheatography.com/jumpingjaden/

Published 26th January, 2025.

Last updated 26th January, 2025.

Page 2 of 3.

Sponsored by [Readable.com](https://readable.com)

Measure your website readability!

<https://readable.com>

std::strings

defining a string	string s1("Man"); or string s2 = "Beast"; or string s3;
combining strings	use the + operator. s3 += "" and "" + s2
Getline	reads up to an enter aka new line getline(cin, name);
cin	reads up to a space cin >> name;
fixes for when coming after a cin	add a cin.ignore(); before the getline statement
.size() and .length()	do the same thing, finds the length of the string
Concatenating with numbers	use to_string. string full = str + to_string(value);
comparing strings	same idea as values A is smaller in askii than B, Lower case values are larger than upper case in the askii table.

std::strings (cont)

.find()	returns the position of the search item.specify a start position of search str.find("iss"); can also look for the option after the first time pos = str.find("iss"); str.find("iss", pos + 1)
string::npos	no position aka not found
substring	returns a portion of the string and specifies the end. int space = str.find(" "); string school = str.substr(0, space);
.at vs [] for iterating over strings	.at is read only where as [] is read and write, .at will terminate if you are looking for something out of range [] will display random junk
cstring vs std::string	c-string pros: fixed length and primitive data type. cons: watch for out of bounds. std::strings more intuitive to use and checks for out of bounds.

stringstream

#include <sstream>	adds strings, objects and values into a stream, parse streams and strings into values
<<	used to add to the stringstream. .str() function converts the stream to a string. stringstream sout; sout << "add words";
parsing	string str = "one two three four"; stringstream sout; sout << str; string temp; while(sout >> temp) {cout << temp << endl;}

class member functions example

```

class Student
{
private:
    int id;
    string name;
    string email;

public:
    //constructor
    Student(int id, name, email) {}
    Student(int id, string name, string email) : id(id), name(name), email(email) {}

    //getter
    void GetID() const { return id; }
    void SetName(string name) { name = name; }
    void SetEmail(string email) { email = email; }

    //setter
    int GetID() const { return id; }
    string GetEmail() const { return email; }
};
    
```

example main, cpp and h files

```

// Student.h
#pragma once
#include <string>
using namespace std;

class Student {
public:
    Student(int id, string name, string email) : id(id), name(name), email(email) {}
    Student(int id, string name, string email) : id(id), name(name), email(email) {}

    //getter
    void GetID() const { return id; }
    void SetName(string name) { name = name; }
    void SetEmail(string email) { email = email; }

    //setter
    int GetID() const { return id; }
    string GetEmail() const { return email; }
};

// Student.cpp
#include "Student.h"

//constructor
Student::Student(int id, string name, string email) : id(id), name(name), email(email) {}
Student::Student(int id, string name, string email) : id(id), name(name), email(email) {}

//getter
void Student::GetID() const { return id; }
void Student::SetName(string name) { name = name; }
void Student::SetEmail(string email) { email = email; }

//setter
int Student::GetID() const { return id; }
string Student::GetEmail() const { return email; }

//main.cpp
#include "Student.h"
using namespace std;

int main() {
    Student s1(1, "John", "john@example.com");
    Student s2(2, "Jane", "jane@example.com");

    s1.SetName("John D.");
    s1.SetEmail("john.d@example.com");

    s2.SetName("Jane D.");
    s2.SetEmail("jane.d@example.com");

    cout << s1.GetID() << endl;
    cout << s1.GetEmail() << endl;

    cout << s2.GetID() << endl;
    cout << s2.GetEmail() << endl;

    return 0;
}
    
```

parts of a function

