CPP SYNTAX & BASICS

BASIC PROGRAM

#include <iostream>

Using namespace std;

Int main() {

cout << “Hello World” << endl; // endl = endline, (or use \n)

return 0;

}

**Compile:** g++ main.cpp -o run

DATA TYPES

|  |  |  |
| --- | --- | --- |
| **Primitive** | **Derived** | **Abstract** |
| Int | Function | Class |
| Char | Array | Structure |
| Bool | Pointer | Union |
| Float | reference | Enum |
| Double |  | Typedef |
| Void |  |  |
| Wide char |  |  |

BASIC CALCULATOR

Int main() {

Int a, b, sum;

Cout << “Enter a number: \n”;

Cin >> a;

Cout << “Enter another number: “ << endl;

Cin >> b;

Sum = a + b;

Cout << “The sum of “ << a << “ and << b << “ is ” << sum << endl;

Return 0;

}

CLASSES & OBJECTS

Basic Example

Class Animal {

Public: // access. Other: protected, private

Void eat() {

Cout << “The animal is eating”;

};

};

Int main() {

Animal animalObject; // local – stack. Deleted when out scope

animalObject.eat();

Animal otherAnimal = new Animal; // pointer – must delete to prevent leak

// allocates memory on the *free store*

otherAnimal->eat();

}

Constructor & Destructor

A constructor is a method called when class is instantiated. In order for a method to be a constructor, it must be named the same as the class name.

Class Animal {

Public: // public access

Animal() {cout << “constructed”;} // no data type

~Animal() {cout << “deleted”;} // notice ~tilde (destructor)

}

Inherit parent constructor

Class Species {

Protected:

String name;

Public:

Species(string n) :name(n) {};

};

Class Human : public Species {

Private:

Int age;

Public:

Human(string n, int a)

: Species(n), age(a) // call parent construct and set property

{

// code

};

}

Access

Class Animal {

Private:

Int age;

Public:

Animal(int a)

: age(a) // the correct way to initialise members

{

Cout << “Animal constructed”;

};

Void eat(string food = “lamb”) { // code } // default argument

}

Inheritance

Class Dog : public Animal {}

The Animal constructor will fire first, as the base classes must all be set up prior to being able to use an instance of the Dog class.

Constants

Class Person {

Private:

Const string className = “Person”;

Public:

String getClassName() **const** {

Return this->className;

}

};

Int main() {

Const Person ali;

Cout << ali.getClassName() << endl; // would error if getClassName not const

}

This

*This* stores the address of the current object you are working with. In other words, it is a pointer so you need to call members using -> syntax:

This->memberName;

(\*this).memberName; // or dereference pointer

Cout << this << endl; // will print out hex address

Virtual function/method

An inheritable and overridable function.

Void func(Animal \*animal) { animal->eat(); };

Class Animal {

Public:

Virtual void eat() { cout << “eating generic food”; };

}

Class Cat : public Animal {

Public:

Void eat() { cout << “Eating rat”; };

}

Int main() {

Animal \*animal = new Animal;

Cat \*cat = new Animal;

Func(animal);

Func(cat); // as func takes Animal data type, this would have outputted Animal::eat() if it wasn’t virtual

}

In other words, it searches for the correct derived class to see if virtual method has been overridden. If not labelled virtual then it will output the first method it comes to.

Abstract class

An abstract class is any class with a pure virtual function in it. A virtual function is written as so:

Virtual void print() = 0;

This *must* be overridden in the derived class! You cannot instantiate abstract classes! You If no pure virtual functions present then it is not an abstract class, and you can instantiate it, (see above section).

HEADER FILES / SEPERATING CLASSES

Must include a cpp file and a header file

*Animal.h*

#ifndef ANIMAL\_H\_

#define ANIMAL\_H\_

Class Animal {

Public:

Animal(); // constructor

Void eat(); // no functionality

}

#endif

*Animal.cpp*

#include “Animal.h” // must include header file

#include <iostream>

Using namespace std;

Animal::Animal() {

Cout << “constructed”:

};

Animal::eat() { // :: binary scope resolution operator

// functionality here

}

// remember when compiling to also compile all other cpp files used by the system

FUNCTION OVERLOADING

CPP has a feature where 2 or more functions can exist with the same name but have different parameters:

1. Void printNumber(**int** x) { cout << x << endl; };
2. Void printNumber(**double** x) {cout << x << endl };

Int main() {

Int a = 12;

Double b = 12.123;

printNumber(a); // will use 1

printNumber(b); // will use 2

}

MULTIDIMENSIONAL ARRAYS

Int grid[2][3] = { // [row][column]

{10, 20, 30},

{40, 50, 60}

};

//print all

For (int I = 0; I < 2; i++) {

For (int j = 0; j < 3; j++) {

Cout << grid[i][j] << ‘,’;

}

Cout << endl;

}

POINTERS

Int num = 5;

Cout << &num << endl; // print address

Int num = 5;

Int \*addr = &num;

Cout << \*addr << “ lives at “ << addr << endl; // value lives at hex address

MEMORY ALLOCATION

Unlike in C, when you are using malloc you must typecast as by default malloc returns void\*

e.g. if allocating memory to a user defined *Person* struct:

Person \*ali = (Person \*) malloc(sizeof(Person));

EXCEPTIONS

Try {

If (//) {

Throw 99;

}

} catch (int x) {

Cout << “Message: “ << x << endl;

}

STRINGS

#include <string> // in order to use string data type

String name = “Ali Issaee”;

Cout << name << endl;

String name;

Cin >> name; // store user input – this will only store up to first white space

Getline(cin, name); // store user input up to first line

FILES

#include <fstream>

Write to file

Int main() {

Ofstream myFile; // ofstream – write to file

myFile.open(“file.txt”); // can also pass in constructor on line above

myFile << “Hello World” << endl; // write to file

myFile.close();

}

Check if file is open with: *myFile.is\_open();*

int main() {

ofstream myFile(“file.txt”);

cout << “Enter ID, name and money”;

int id;

string name;

double money;

while (cin >> id >> name >> money) {

myFile << id << ‘ ‘ << name << ‘ ‘ << money << endl;

}

}

// the while loop will end when user exits from program (ctrl + z)

Read from file

Int main() {

Ifstream myFile(“file.txt”); // ifstream – read from file

Int id;

String name;

Double money;

While (myFile >> id >> name >> money) {

Cout << id << ‘, ‘ << name << ‘, ‘ << money << endl;

}

}

// the while loop will end when EOF reached. The file will also automatically close in the destructor (when EOF reached), so no need to close the file manually.

**QUIZ**

PART 1

1. Create a ‘Hello World’ program
2. Add 2 numbers together, retrieved from user input, and output the answer
3. Build a class with private and public methods
4. Instantiate above class with and without *new* keyword – what’s the difference?
5. Create a new class and inherit from it
6. Give the above class a constructor and destructor
7. Initialise the members in the above class the ‘right way’

PART 2

1. Create a class with constant methods, and call them using constant object
2. What is *this* – give example
3. What is a virtual function – give example
4. Create an abstract class and inherit from it
5. Create a new class and put it in a separate file (cpp and header file), and use it in main

PART 3

1. What is function overloading – give example
2. Print a multi-dimensional array to the terminal
3. Allocate memory to a struct using malloc
4. Give example of how to use exceptions in cpp
5. Write data to a file retrieved from user input
6. Read from a file and output to a terminal