#include <iostream> (Input output stream)

#include <string> (String data)

#include <fstream> (File association)

Declaration : ofStream outputFIle (Output to file), ifstream inputFile (Input from fileMethods: outputFile.open(text), outputFile.close()) **[Use double slashes "\\" to specify path]**

Filename variable cannot be string if used to open file. Use char filename, or filename.c_str()

Using namespace std;

Int main()

{

String var1, var 2; (Both are strings)

Cout << msg;

Cin >> var;

Getline(input method (cin, inFile), var);

}

Variable types

Numbers (int,long (at least 2x more data),double)

String var

Char var[number]

Bool var

If statements

Statements can be covered with {}.

If (expression)

{Statement};

Be careful when comparing double variables.

If expression is a number larger than 1, the expression is true.

if (expression) statement; elseif (expression) statement;

else statement; (else does not require a ";" after if's {})

if (s1 OR s2) , **ONLY EVALUATES s1 if s1 is true (s2 is NOT EVALUATED).**

&& AND, || OR, ! NOT.

Loop

while (expression) statement;

for (initialization, test; update) {statement};

e.g. for(int count = 1; count <=5; count++ or count += n)

Infinite loop goes up and down.

Do statement; while (expression) -> **Post-test loop. Do once, then checks for loop condition.**

InFile

Ifstream infile;

Infile.open("myFile.txt");

While(!infile.eof()) {infile >> data} **While infile has not reached end of data, store the file data into a variable.**

Or while(infile >> data) {} **While data is being stored into a variable, program is reading the file.**

Arrays

Int testArray[5]; [5] is the number of elements in the array. This means it can hold 5 integers. **C++ reserves blocks of memory for the array.** The size is (num of elements) * (number of bytes for each element)

Array[0] = 10; **Same as python. Subscript can be a literal, variable, or expression.**

Const int SIZE = 5; int firstArray[SIZE] = {1,2,3,4,5}

Int secondArray[SIZE] = {5,10}; **Partial initialization, the rest of the list is 0.**

Int thirdArray[] = {1,2,3,4,5} **Implicit initialization**

C++ does not check bounds.

Two dimensional array: int exams[ROWS][COLS]

Functions

Void func(int a) { statements } -> **Void function (Function that does not return a value)**

Int main() { statements, return 0 } -> **Main function that returns 0 at the end**

Function must be declared before main function (Function not declared in this scope)

Function Prototype: Function definition without body (e.g. var Func1(); before main function)

Function variables are local to their scope. Do not use Global variables.

Pass by value: Value is copied into function, instead of the variable.

Pass by reference: Change in parameter also made to argument. Use & before var in parameter. (e.g. int &a). Only variables can be used. Can create debugging problems. Array is automatically passed by referece.

& also used as address of operator. Displays a variable address in terms of hex.

Pointers: Variables that holds a memory address. Can specify where to store data in RAM. RAM is in increment of 4 bytes.

Pointers can point to following pointers. XD

Code about Pointers

***var <- Creates a pointer variable. (Used when initializing)**

*** is also a dereferencing operator. Converts hex back to integer. (Used during the rest of the code)**

When there is a * in front of the variable, the value it returns should be an integer. Otherwise, the value returned should be hex (0xsomething)

Do not store var into star. Initialize the variable without using "*" .

Can set pointers to arrays, but cannot set arrays to pointers.

e.g. ptr = arr is correct, but arr = ptr is wrong.

+1 Pointer -> +1 Sizeof data type

Segmentation fault/bus error/access violation -> Tried to access memory that machine cannot address.

Another way to do pass by reference

Use pointers. & in arguments, and * in parameters.

Cannot enter memory location through keyboard

2 ways to find max in array using pointers:

- Set *array to max, array ++, if *array > max, max = *array
- Set *array to max, I++, if *(array + i) > max, max = *(array + i)

New operator

Double *array;

Array = new double array[2];

Memory Leak -> Cannot release memory that has been consumed.

Struct

Struct NAME

{

Declare data here

};

NAME instance = { Values of variables in struct }<- declares instance of struct.

Cannot output struct. Only can output its individual fields.

Structs can be used in functions.

Complex structs can slow down computer.

Can combine arrays and structs.

Struct within struct.

(*s).name == s->name

Class

Class rectangle

{

Private:

varDeclrataion;

functionPrototype;

Public:

varDeclaration;

functionPrototype;

}

Private by default. Fields should be private

Accessor -> Get info from private data, put const at the end

Mutator -> Able to change the value

Default constructor = No arguments

.h file is used to initiate class

Constructor is sued to initialize data fields. [Class(v1,v2,v3)]

Destructor -> ~object, delete in main function calls the destructor.

Dynamically memory allocated variable should be deleted at the end of the function.

Private -> Only accessible within its own class. Subclass cannot access it

Protected -> Private but accessible by objects of derived class (Subclass can access it)

Public -> All can access.

Private -> Protected -> Public

Static -> Compile Time

Virtual -> Runtime (Use this for polymorphism)

Abstract class e.g. Animal.sound()

Pure Virtual Function

Virtual -> Do not make any assumptions.

If there is a virtual keyword, look at the latest way it is defined.

Data Stuff

O(n) time complexity

N is the cause of the time complexity.

O(1) -> Constant time

First/Last will always end up next to middle

Binary search for 100000 -> 17

Log function (O(log n))

Bubble sort and Selection Sort both have time complexity O(n^2), but selection sort is faster if n is small.

Linked List: Head (pointer to next) -> Data1 (pointer to next) -> Data2 (pointer to next) -> …. -> Null

Linked lists -> Sequential access, Arrays -> Direct(Random) Access

Change the pointer first, then delete its current

Template <class T>

T is a flexible var type

Vector -> Expandable array

Functions:

Push_back() -> Appends element to the vector

Capacity() -> Memory stuff

Pop_back() -> Deletes the last element in vector

Empty() -> Returns Boolean if empty

Clear() -> Clears the vector

Begin() -> First element, End() -> Space next to last element -> Returns iterator (pointer)

Iterator -> Pointer

Vector::vector

For(iter = vect.begin(); iter!=vect.end; iter++) cout << *iter << " ";

Vect.insert(iterator,element) Note: Inserts an element before the iterator, but iterator is unchanged

Doubly linked list:

Head -> Node 1 -> Node 2 -> Node 3 -> Null

Tail -> Node 3 -> Node 2 -> Node 1 -> Null

Vector has almost same instructions as a doubly linked list, but inserting a full vector might take more time since it has to create a new vector, deconstruct and reconstruct.

Stacks -> Just like putting dishes in the container.

Stack can be applied to Matching Brackets.

e.g. ([]) is legal because (-> [->] ->)

Pop the top if equal bracket, leave if not.

If there are no brackets then it matches.

If there are no items in the stack then it matches.

Queue -> Two pointers. Front and rear.

Dequeue removes the next element. Enqueue puts a value into the current spot.

Recursion works like a stack.

Binary Tree

Child nodes -> What is below the node

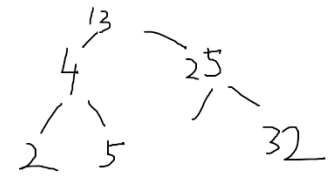
Parent nodes -> What is above the node

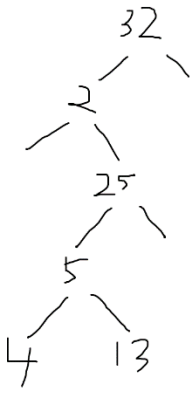
Leaf nodes -> No children for left and right

Subtree -> Portion from node down to leaves

Only nodes can be parents

Inserting:





Pointers can also be passed by reference or by value.

By value only changes the memory address inside of the function. E.g. *a works in and out, while a only works within