Day 1

File Handling

Opening a FILE

A file must be opened before you can read from it or write to it.Either ofstream or fstream object may be used to open a file for writing. And ifstream object is used to open a file for reading purpose only.

Syntax : void open(const char \*filename, ios::openmode mode);

* The first argument specifies the name and location of the file to be opened and the second argument of the open() member function defines the mode in which the file should be opened.the second argument is optional.

ios::app - append mode

ios::ate - move the read/write control to the end of the file

ios::in – reading mode

ios::out – writing mode

ios::trunc - If the file already exists, its contents will be truncated before opening the

file.

Closing a File

Syntax : void close();

## Writing to a File

While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an **ofstream** or **fstream** object instead of the **cout** object.

## Reading from a File

You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an **ifstream** or **fstream** object instead of the **cin** object.

Example:

#include <fstream>

#include <iostream>

using namespace std;

int main () {

char data[100];

ofstream outfile;

outfile.open("afile.txt");

cout << "Writing to the file" << endl;

cout << "Enter your name: ";

cin.getline(data, 100);

outfile << data << endl;

cout << "Enter your age: ";

cin >> data;

outfile << data << endl;

outfile.close();

ifstream infile;

infile.open("afile.txt");

cout << "Reading from the file" << endl;

infile >> data;

cout << data << endl;

infile >> data;

cout << data << endl;

infile.close();

return 0;

}

Dynamic Memory Allocation

NEW:

The new operator is used to allocate the memory for the called variable at the runtime within the heap for the variable of a given type, which returns the address of the space allocated.

Syntax : new data-type;

data-type could be any built-in data type including an array or any user defined data types include class or structure

Delete:

a variable that has been dynamically allocated is not anymore required , one can free up the memory that it occupies in the free store with the ‘delete’ operator.

Templates:

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

Function templates:

Syntax:

template <class type> ret-type func-name(parameter list) {

// body of function

}

Example:

#include <iostream>

#include <string>

using namespace std;

template <typename T>

T const& Max (T const& a, T const& b) {

return a < b ? b:a;

}

int main () {

int i = 39;

int j = 20;

cout << "Max(i, j): " << Max(i, j) << endl;

double f1 = 13.5;

double f2 = 20.7;

cout << "Max(f1, f2): " << Max(f1, f2) << endl;

string s1 = "Hello";

string s2 = "World";

cout << "Max(s1, s2): " << Max(s1, s2) << endl;

return 0;

}

Class Template

Syntax:

template <class T>

class className {

private:

T var;

public:

T functionName(T arg);

};

Example:

#include <iostream>

using namespace std;

template <class T>

class Number {

private:

// Variable of type T

T num;

public:

Number(T n) : num(n) {}

T getNum() {

return num;

}

};

int main() {

Number<int> numberInt(7);

Number<double> numberDouble(7.7);

cout << "int Number = " << numberInt.getNum() << endl;

cout << "double Number = " << numberDouble.getNum() << endl;

return 0;

}