@File handling

File handling in C++ involves reading from and writing to files using standard libraries. Here's a basic overview:

1. Including the Required Header

To work with files, you need to include the <fstream> header, which provides facilities for file handling:

#include <fstream>

2. File Streams

C++ provides three main file stream classes:

ifstream (input file stream) for reading from files.

ofstream (output file stream) for writing to files.

fstream (file stream) for both reading and writing.

1 
2 
3 
4 
5 
6 
7 
8 
9 
lø 
11 
12 
13 
14 
15 
*inc lude<i0St ream> 
using namespace std; 
int main() 
// File ko open karna 
ofstream tout; 
fout. open( "zoom. txt" ) ; 
// write kar sakta hu 
India"; 
// create kar dega aur fr open kar deg 
// Resource release kar paauon 

2 
3 
4 
5 
6 
7 
8 
9 
lø 
11 
12 
13 
14 
15 
16 
17 
18 
19 
2ø 
21 
22 
# inc 
# include<fst ream> 
using namespace std; 
int main() 
the input: • 
for( int 
// file ko open karo 
ofstream fout; 
tout. open( " zero. txt"); 
for( int 
fout«arrtil«•• 
tout. ) ; 

3. Opening and Closing Files

You open a file by creating an instance of the file stream classand specifying the file name. You can also specify the mode (e.g., input, output, append).

Opening a file:

std::ifstream inputFile("input.txt"); // Open for reading

std::ofstream outputFile("output.txt"); // Open for writing

std::fstream file("file.txt", std::ios::in | std::ios::out); // Open for both reading and writing

Closing a file:

cpp

Copy code

inputFile.close();

outputFile.close();

file.close();

4. Reading from Files

You can read from a file using the >> operator or various methods such as getline():

cpp

Copy code

std::string line;

std::ifstream inputFile("input.txt");

while (getline(inputFile, line)) {

std::cout << line << std::endl; // Print each line

}

inputFile.close();

5. Writing to Files

You write to a file using the << operator:

cpp

Copy code

std::ofstream outputFile("output.txt");

outputFile << "Hello, World!" << std::endl;

outputFile << "This is a test." << std::endl;

outputFile.close();

6. File Modes

When opening a file, you can specify different modes using flags such as:

std::ios::in: Open for reading.

std::ios::out: Open for writing.

std::ios::app: Append to the end of the file.

std::ios::ate: Move to the end of the file when opening.

std::ios::trunc: Truncate the file to zero length.

Example:

cpp

Copy code

std::ofstream outputFile("output.txt", std::ios::app); // Open in append mode

7. Error Handling

Always check if a file opened successfully:

cpp

Copy code

std::ifstream inputFile("input.txt");

if (!inputFile) {

std::cerr << "Error opening file!" << std::endl;

return 1;

}

8. File Pointers

You can use file pointers to manipulate the file position:

cpp

Copy code

std::fstream file("file.txt");

file.seekg(0, std::ios::end); // Move to the end of the file

std::streampos fileSize = file.tellg(); // Get file size

file.seekg(0, std::ios::beg); // Move back to the beginning

9. Binary Files

For binary files, you use std::ios::binary mode:

Writing binary data:

cpp

Copy code

std::ofstream outputFile("binary.dat", std::ios::binary);

int num = 123;

outputFile.write(reinterpret\_cast<char\*>(&num), sizeof(num));

outputFile.close();

Reading binary data:

cpp

Copy code

std::ifstream inputFile("binary.dat", std::ios::binary);

int num;

inputFile.read(reinterpret\_cast<char\*>(&num), sizeof(num));

std::cout << num << std::endl;

inputFile.close();

This covers the basics of file handling in C++. If you have specific questions or need examples, feel free to ask!