**Practical 11 – refer to Topics20 and 21**

**Part A(Understanding Concepts)**

1. A sample of the input and output file is as follows:

|  |  |
| --- | --- |
|  |  |

And, given the following declarations:

ifstream inFile;

ofstream outFile;

int number1, number2, result;

1. Write statements to open a file named “mydata.txt” for reading and associate it with file stream inFile and to open a file named “myreport.txt” for writing and associate it with file stream outFile. Check for any errors when opening the files.

Ifstream inFile (“mydata.txt”);

If (!inFile)

Cout << “error opening file”  
ofstream outFile(“myreport.txt”);  
if (!outFile)  
Cout << “error opening file”

Write statements to read a two numbers from “mydata.txt” into 2 variables called number1 and number2, compute their sum and store into a variable called result, and write the result to the file “myreport.txt”.  
Ifstream inFile (“mydata.txt”);

If (!inFile)

Cout << “error opening file”

Else  
{ // reading numbers from the file

cout << "The numbers from the file are:\n";

inFile >> number1 >> number2;

}

ofstream outFile(“myreport.txt”);  
if (!outFile)  
Cout << “error opening file”  
while (outFile)

{

cout << number1 << “ + “ << number 2 << endl;

sum= number1 + number2; // cal. "sum"

outFile <<” = =” << sum <<endl;

}

1. Write statements to close the two files.  
   inFile.close();  
   outFile.close();
2. Write a program to read the lines of text from one text file and write them to another text file with all the letters changed to uppercase. Assume each line in the file does not exceed 50 characters. Note: use getline() to read a line containing spaces.

A sample of the input and output file is as follows:

Input file contents Output file contents

AN APPLE A DAY KEEPS THE DOCTOR AWAY.

A STITCH IN TIME SAVES NINE.

MORE HASTE LESS SPEED.

An apple a day keeps the doctor away.

A stitch in time saves nine.

More haste less speed.

**Part B (Programming Exercises)**

1. Write a program to read a set of numbers from a file (“numbers.txt”), compute and display the product of the numbers on the screen.

#include <iostream>

#include <fstream> // preprocessor for file handling

using namespace std;

int main(void)

{

int number;

long product = 1; // declare an initialise for "product"

ifstream inFile("numbers.txt"); //open file

if (!inFile) // check for error in opening file

cout << "Error opening file\n";

else

{ // reading numbers from the file

cout << "The numbers from the file are:\n";

inFile >> number;

while (inFile)

{

cout << number << endl;

product \*= number; // cal. "product"

inFile >> number;

}

cout << "Product = " << product << endl; // print result

inFile.close(); //close the file

}

return 0;

}

1. Write a program to read a set of numbers from a file, store them in an array, and display the numbers in reverse order. Assume there are no more than 30 numbers in the file.

#include <iostream>

#include <iomanip>

#include <fstream> // preprocessor for file handling

using namespace std;

int main(void)

{

ifstream inFile("numbers.txt"); //open file

if (!inFile) // check for error in opening file

cout << "Error opening file\n";

else

{ // reading numbers from the file into an array

int numbers[30], index = 0;

inFile >> numbers[index]; //read first number

while (inFile)

{

index++;

inFile >> numbers[index]; //read next number

}

// print array in reverse order

for (int i = index - 1; i >= 0; i--)

{

cout << setw(4) << numbers[i];

if ((i - index) % 5 == 0) // 5 numbers per line

cout << endl;

}

inFile.close(); //close the file

}

return 0;

}

1. Write a program to input a set of words from the user and write the words to a file with spaces added in between the letters. Assume the longest word is 20 characters long. A sample input and output is as follows:

Input/Outputusing keyboard/screen Output file contents

Enter a word: Apple

A p p l e

B a l l

C h a i r

D o g

E l e p h a n t

Enter a word: Ball

Enter a word: Chair

Enter a word: Dog

Enter a word: Elephant

Enter a word: ^Z

#include <iostream>

#include <iomanip>

#include <fstream> // preprocessor for file handling

using namespace std;

int main(void)

{

ofstream outFile("words.txt"); //open file

if (!outFile) // check for error in opening file

cout << "Error opening file\n";

else

{

char word[21];

cout << "Enter a word: "; //read first word

cin >> word;

while (cin)

{

for (int i = 0; word[i] != '\0'; i++)

outFile << word[i] << " ";

outFile << endl;

cout << "Enter a word: "; //read first word

cin >> word;

}

outFile.close(); //close the file

}

return 0;

}

1. Write a program to read an input file containing student data. Each line of text in the file contains a student’s registration number and 3 test scores. The program computes the average score.

Use a structure to represent a student. The structure has 5 members to store the registration number (string), 3 test scores (array of floats), and average score (float). The program should be able to handle a maximum of 50 students using an array of structures. The program produces an output file with a report showing, for each student, the registration number, 3 tests scores, and the average score.

A sample of the input and output file is as follows:

Input file contents Output file contents

Reg. No. Test1 Test2 Test3 Average

UA1001 82.0 91.0 76.0 83.0

UA1002 77.0 54.0 61.5 64.2

UA1003 65.5 63.0 62.0 63.5

UA1001 82.0 91.0 76.0

UA1002 77.0 54.0 61.5

UA1003 65.5 63.0 62.0

#include <iostream>

#include <iomanip>

#include <fstream>

using namespace std;

typedef struct

{

char number[7];

float scores[3];

float average;

} STUD\_TYPE;

int main(void)

{

ifstream in\_file("tests.txt");

ofstream out\_file("test\_report.txt");

if (!in\_file || !out\_file)

{

if (!in\_file)

cout << "Error in opening input file.\n";

if (!out\_file)

cout << "Error in opening output file.\n";

}

else

{

STUD\_TYPE students[50];

int index = 0;

float total;

in\_file >> students[index].number;

while (in\_file)

{

total = 0;

for (int i = 0; i < 3; i++)

{

in\_file >> students[index].scores[i];

total += students[index].scores[i];

}

}

students[index].average = total / 3;

index++;

in\_file >> students[index].number;

//output to text file

out\_file << "Reg.No. Test1 Test2 Test3 Average\n";

cout << fixed << setprecision(2);

for (int i = 0; i < index; i++)

{

out\_file << students[i].number << " ";

for (int j = 0; j < 3; j++)

{

out\_file << setw(6) << students[i].scores[j];

out\_file << setw(7) << students[i].average << endl;

}

}

in\_file.close();

out\_file.close();

}

return 0;

}

**Part C (Self-Review / Revision)**

1. What must be done before a file can be used for reading or writing? after we have finished using it?
2. What are the different file modes that can be used when opening a file?
3. What are the functions that can be used to read data from a file? write data to a file?

**Part D (Practice Exercises)**

1. Write and test a program that reads product details from a file and stores the information in an array of structures. The program then prints a menu to allow the user to do any of the following:

1. List all products

2. Search the price of a product

3. Update the price of a product

4. Exit

If the user chooses to exit, the program will write the updated data in the array of structures to the file.

Product information includes product number, description and price.

#include <iostream>

#include <iomanip>

#include <fstream>

#include <cstring>

using namespace std;

typedef struct

{

char number[6];

char desc[31];

double price;

} PRODUCT\_TYPE;

void list (PRODUCT\_TYPE prod[], int rows);

void show\_price (PRODUCT\_TYPE prod[], int rows);

void update\_price (PRODUCT\_TYPE prod[], int rows);

int search\_product(PRODUCT\_TYPE prod[], int rows);

int main()

{

ifstream in\_file("product.txt");

if(!in\_file)

cout << "Error opening input file\n";

else

{

PRODUCT\_TYPE products[50];

int index = -1, choice;

in\_file.getline(products[++index].number, 6);

while(in\_file)

{

in\_file.getline(products[index].desc, 31);

in\_file >> products[index].price;

// clear unwanted whitespace

if(in\_file.peek() == '\n')

in\_file.ignore(256, '\n');

// read next number

in\_file.getline(products[++index].number, 6);

}

in\_file.close();

// menu starts

do

{

cout << "Do you want to:\n";

cout << "1. List all products\n";

cout << "2. Display the price for a product\n";

cout << "3. Update the price for a product\n";

cout << "4. Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice)

{

case 1 : list(products, index); break;

case 2 : show\_price(products, index); break;

case 3 : update\_price(products, index); break;

case 4 : break;

default: cout << "Invalid choice\n"; break;

}

} while (choice != 4);

ofstream out\_file("product.txt");

if(!out\_file)

cout << "Error opening output file, records are not updated.\n";

else

{

for (int i = 0; i < index; i++)

{

out\_file << products[i].number << endl;

out\_file << products[i].desc << endl;

out\_file << products[i].price << endl;

}

out\_file.close();

}

}

return 0;

}

void list(PRODUCT\_TYPE prod[], int rows)

{

cout << fixed << setprecision(2);

for (int i = 0; i < rows; i++)

cout << prod[i].number << "\t" << prod[i].desc << "\t"

<< setw(8) << prod[i].price << endl;

return;

}

int search\_product(PRODUCT\_TYPE prod[], int rows)

{

int i = 0;

bool found = false;

char number[6];

cout << "Enter product number to search: ";

fflush(stdin);

cin.getline(number, 6);

while (i < rows && !found)

{

if (strcmp(number, prod[i].number) == 0)

found = true;

else

i++;

}

if (found)

return i;

else

return -1;

}

void show\_price(PRODUCT\_TYPE prod[], int rows)

{

int pos = search\_product(prod, rows);

if (pos != -1)

cout << "Current price is " << prod[pos].price << endl;

else

cout << "Product not found\n";

return;

}

void update\_price (PRODUCT\_TYPE prod[], int rows)

{

int pos = search\_product(prod, rows);

if (pos != -1)

{

double newPrice;

cout << "Current price is " << fixed << setprecision(2) << prod[pos].price;

cout << "\nEnter new price: ";

cin >> newPrice;

prod[pos].price = newPrice;

}

else

cout << "Product not found\n";

return;

}