**MC5211 – OBJECT ORIENTED PROGRAMMING LABORATORY**

**LAB MANUAL**

**DAY OF WEEK USING ENUMERATION TYPE**

**EX NO.**: 1((a)

**DATE:**

**Aim**: To write a C++ program to display the day of week using enum type.

**Algorithm:**

Step 1: Start.

Step 2: define the enum type for week.

Step 3: Declare variable using enum.

Step 4: Compute the day of week.

Step 5: Display the result.

Step 6: Stop.

**Program:**

// Program for displaying day of week using enum type

#include<iostream.h>

Using namespace std;

Enum week { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};

Int main()

{

week today;

today = Wednesday;

cout << “ day order of “ << today << “ is “ << today+1;

return 0;

}

**Output**

**Day order of Wednesday is 4**

**OVERLOADED FUNCTION**

**EX NO.**: 1((b)

**DATE:**

**Aim**: To write a C++ program to calculate square of the given input using overloaded function.

**Algorithm:**

Step 1: Start.

Step 2: Declare two function using the name “Square” with different typeofparameters

Step 3: Call the function Square with different type of values ( e.g. Square(7), Square(7.5).

Step 4: Display the result.

Step 5: Stop.

**Program:**

// Program for calculating square using overloaded functions

#include <iostream>

using std::cout;

using std::endl;

int square( int x ) { return x \* x; }

double square( double y ) { return y \* y; }

int main()

{

cout << "The square of integer 7 is " << square( 7 )

<< "\nThe square of double 7.5 is " << square( 7.5 )

<< endl;

return 0;

} // end function main

**Output:**

The square of integer 7 is 49

The square of double 7.5 is 56.25

**SCOPE OF VARIABLES AND STORAGE CLASSES**

**EX NO.**: 2(a)

**DATE:**

**Aim**: To write a C++ program to declare and display different local and Global variables.

**Algorithm:**

Step 1: Start.

Step 2: Declare variables of local (within function) and global scopes(outside main).

Step 3: Display the variables’ content using function.

Step 5: Stop.

**Program:**

// Program for **Local variable and Global variable and its usage,**

#include <iostream>

using namespace std;

// Global variable declaration

int c = 12;

void test();

int main()

{

// local variable to main()

int var = 5;

++c;

// Outputs 13

cout << c <<endl;

test();

// illegal: var1 not declared inside main()

var1 = 9;

}

void test()

{

// local variable to test()

int var1;

var1 = 6;

// illegal: var not declared inside test()

cout << var;

++c;

// Outputs 14

cout << c;

}

**Output**

13

14

**STORAGE CLASSES**

**EX NO.**: 2(b)

**DATE:**

**Aim**: To write a C++ program to declare different local and Global variables with different scope levels and display the contents.

**Algorithm:**

Step 1: Start.

Step 2: Declare variables of local (within function) and global scopes (outside main) with different levels of scope.

Step 3: Display the variables’ content using function.

Step 5: Stop.

**Program:**

// Program for Static local variable

#include <iostream>

using namespace std;

void test()

{ // var is a static variable

static int var1 = 0;

++var1;

cout << var1 << endl;

// local variable to main()

auto int var2 = 5;

cout << var1 << endl; }

int main()

{

test();

test();

return 0; }

**Output:**

1

5

2

5

**ITERATORS USING VECTORS**

**EX No. : 9a**

**DATE:**

**AIM:**

To write a C++ program to implement a iterator using Vectors.

**ALGORITHM:**

1. Start

2. Get the values

3. Assign the iterator

4. Display the result

5. Stop

**SOURCE CODE:**

// C++ code to demonstrate the working of // iterator, begin() and end()

#include<iostream>

#include<iterator> // for iterators

#include<vector> // for vectors

using namespace std;

int main()

{

    vector<int> ar = { 1, 2, 3, 4, 5 };

    // Declaring iterator to a vector

    vector<int>::iterator ptr;

    // Displaying vector elements using begin() and end()

    cout << "The vector elements are : ";

    for (ptr = ar.begin(); ptr < ar.end(); ptr++)

        cout << \*ptr << " ";

    return 0;

}

**Output:**

The vector elements are: 1 2 3 4 5

**FILE HANDLING-SEQUENTIAL ACCESS**

**EX No. : 9a**

**DATE:**

**AIM:**

To write a C++ program to implement a file handling concept using sequential access.

**ALGORITHM:**

1. Start

2. Get the input string

3. Write the input string character by character into a file called “sample” using put() function

4. Read the input string character by character from the file called “sample” using get() function

5. Display the result

**SOURCE CODE:**

/\* C++ Sequential Input and Output Operations with Files \*/

#include<iostream.h>

#include<fstream.h>

#include<string.h>

#include<stdlib.h>

#include<conio.h>

struct customer

{

char name[20];

float balance;

};

void main()

{

//clrscr();

customer savac;

cout<<"Enter your name: ";

cin.get(savac.name, 20);

cout<<"Enter balance: ";

cin>>savac.balance;

ofstream fout;

fout.open("Saving", ios::out | ios::binary); // open output file

if(!fout)

{

cout<<"File can\'t be opened..!!\n";

cout<<"Press any key to exit...\n";

getch();

exit(1);

}

fout.write((char \*) & savac, sizeof(customer)); // write to file

fout.close(); // close connection

// read it back now

ifstream fin;

fin.open("Saving", ios::in | ios::binary); // open input file

fin.read((char \*) & savac, sizeof(customer)); // read structure

cout<<savac.name; // display structure now

cout<<" has the balance amount of Rs. "<<savac.balance<<"\n";

fin.close();

cout<<"\nPress a key to exit...\n";

getch();

}

**OUTPUT:**

Enter your name: John

Enter balance: 10000

John has the balance amount of Rs.10000

**FILE HANDLING-RANDOM ACCESS**

**EX No. : 9b**

**DATE:**

**AIM:**

To implement file handling concept using random access

**ALGORITHM:**

1. Start

2. Get the input file called random.txt and check the file’s presence

3. Seek the input file to a particular(random) location and get the specified output

4. Display the result

**SOURCE CODE:**

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

void main()

{

clrscr();

fstream fp;

char buf[100];

int pos;

fp.open("D:\\random.txt", ios :: out | ios :: ate); // open a file in write mode with 'ate' flag

cout << "\nWriting to a file ... " << endl;

fp << "This is a line" << endl; // write a line to a file

fp << "This is a another line" << endl; // write another line

pos = fp.tellp();

cout << "Current position of put pointer : " << pos << endl;

fp.seekp(-10, ios :: cur); // move the pointer 10 bytes backward from current position

fp << endl << "Writing at a random location ";

fp.seekp(7, ios :: beg); // move the pointer 7 bytes forward from beginning of the file

fp << " Hello World ";

fp.close(); // file write complete

cout << "Writing Complete ... " << endl;

getch();

fp.open("D:\\random.txt", ios :: in | ios :: ate); // open a file in read mode with 'ate' flag

cout << "\nReading from the file ... " << endl;

fp.seekg(0); // move the get pointer to the beginning of the file

while (!fp.eof()) // read all contents till the end of file

{

fp.getline(buf, 100);

cout << buf << endl;

}

pos = fp.tellg();

cout << "\n Current Position of get pointer : " << pos << endl;

getch();

}

**OUTPUT:**

Writing to a file

Current position of put pointer : 40

Writing Complete

Reading from the file

This is Hello World is a anot

Writing at a random location

Current position of gut pointer : 62