Chapter - 5

Problems

1.Refer to the CIRCAREA program in Chapter 2, “C++ Programming Basics.” Write a function called circarea() that finds the area of a circle in a similar way. It should take an argument of type float and return an argument of the same type. Write a main() function that gets a radius value from the user, calls circarea() , and displays the result.

ans:

//start No.1

#include <iostream>

#define PI 3.14

using namespace std;

float circarea(float radius)

{

return PI\*radius\*radius;

}

int main()

{

float radius;

cout<< "Enter radius to calculate the value of Circle: ";

cin>> radius;

cout<< "Area = " << circarea(radius) << endl;

return 0;

}

//end No.1

2.Raising a number n to a power p is the same as multiplying n by itself p times. Write a

function called power() that takes a double value for n and an int value for p , and

returns the result as a double value. Use a default argument of 2 for p , so that if this

argument is omitted, the number n will be squared. Write a main() function that gets val-

ues from the user to test this function.

ans:

//start No.2

#include <iostream>

using namespace std;

double power(double n,int p){

double result = 0;

for(int i = 0; i < p; i++){

result = n\*n;

}

return result;

}

int main()

{

double n;

cout<< "Enter a number: ";

cin>> n;

cout<< "Square of " << n << " is " << power(n, 2) << endl; //in question default valaue of p should be 2, be care of reading the question in exam lovely

return 0;

}

//end No.2

3.Write a function called zeroSmaller() that is passed two int arguments by reference

and then sets the smaller of the two numbers to 0. Write a main() program to exercise

this function.

ans:

//start No.3

#include <iostream>

using namespace std;

void zeroSmaller(int &first, int &sec){

if(first > sec) sec = 0;

else first = 0;

}

int main()

{

int a, b;

cout<< "Enter first number: ";

cin>> a;

cout<< "Enter sec number: ";

cin>> b;

zeroSmaller(a, b);

cout<< "After call the function\nfirst: " << a << endl << "sec: " << b << endl;

return 0;

}

//end No.3

4.Write a function that takes two Distance values as arguments and returns the larger one.

Include a main() program that accepts two Distance values from the user, compares

them, and displays the larger. (See the RETSTRC program for hints.)

ans:

//start No.4

#include <iostream>

using namespace std;

int distance(int first, int sec){

if(first > sec) return first;

else return sec;

}

int main()

{

int a, b;

cout<< "Enter first distance value: ";

cin>> a;

cout<< "Enter sec distance value: ";

cin>> b;

cout<< "The largest distance is "<< distance(a, b) << endl;

return 0;

}

//end No.4

5.Write a function called hms\_to\_secs() that takes three int values—for hours, minutes,

and seconds—as arguments, and returns the equivalent time in seconds (type long ).

Create a program that exercises this function by repeatedly obtaining a time value in

hours, minutes, and seconds from the user (format 12:59:59), calling the function, and

displaying the value of seconds it returns.

ans:

//start No.5

#include <iostream>

using namespace std;

long hms\_to\_secs(int hr, int minutes, int seconds){

return hr\*3600+minutes\*60+seconds;

}

int main() {

int hr, min, sec;

char sign;

cout<< "Enter hour:min:sec -> ";

cin>> hr >> sign >> min >> sign >> sec;

cout<< hr << sign << min << sec << " = " << hms\_to\_secs(hr, min, sec) << " seconds.\n";

return 0;

}

//end No.5

6.Start with the program from Exercise 11 in Chapter 4, “Structures,” which adds two

struct time values. Keep the same functionality, but modify the program so that it uses

two functions. The first, time\_to\_secs() , takes as its only argument a structure of type time , and returns the equivalent in seconds (type long ). The second function, secs\_to\_time(), takes as its only argument a time in seconds (type long ), and returns a

structure of type time .

ans:

//start No.6

#include <iostream>

using namespace std;

struct tim\_e{

int hr;

int min;

int sec;

};

long time\_to\_secs(tim\_e first\_time){

return first\_time.hr\*3600+first\_time.min\*60+first\_time.sec;

}

tim\_e secs\_to\_time(long seconds){

tim\_e sec\_time;

sec\_time.hr = seconds/3600;

sec\_time.min = seconds%3600/60;

sec\_time.sec = seconds%3600%60;

return sec\_time;

}

int main() {

tim\_e inp\_time;

char sign;

cout<< "Enter time in format (hh:mm:ss): ";

cin>> inp\_time.hr >> sign >> inp\_time.min >> sign >> inp\_time.sec;

int seconds = time\_to\_secs(inp\_time);

cout<< inp\_time.hr << sign << inp\_time.min << sign << inp\_time.sec << " = " << seconds << 's'<< endl;

cout<< seconds << "s = " << secs\_to\_time(seconds).hr << sign << secs\_to\_time(seconds).min << sign << secs\_to\_time(seconds).sec << endl;

return 0;

}

//end No.6

7.Start with the power() function of Exercise 2, which works only with type double .

Create a series of overloaded functions with the same name that, in addition to double ,

also work with types char , int , long , and float . Write a main() program that exercises

these overloaded functions with all argument types.

ans:

//start No.7

#include <iostream>

using namespace std;

double power(double n,int p){

double result = 0;

for(int i = 0; i < p; i++){

result = n\*n;

}

return result;

}

char power(char txt, int p)

{

for(int i = 0; i < p-1; i++){

cout<< txt;

}

return txt;

}

int power(int n, int p){

int result = 0;

for(int i = 0; i < p; i++){

result = n\*n;

}

return result;

}

long power(long n, int p){

long result = 0;

for(int i = 0; i < p; i++){

result = n\*n;

}

return result;

}

float power(float n, int p){

float result = 0;

for(int i = 0; i < p; i++){

result = n\*n;

}

return result;

}

int main()

{

double number\_double = 5;

char character = 'a';

int num\_int = 4;

long num\_long = 67;

float num\_float = 34.6;

cout<< "double type of power function - " << power(number\_double, 2) << endl;

cout<< "Char type of power function - " << power(character, 2) << endl;

cout<< "Int type of power function - " << power(num\_int, 2) << endl;

cout<< "Long type of power function - " << power(num\_long, 2) << endl;

cout<< "Float type of power function - " << power(num\_float, 2) << endl;

return 0;

}

//end No.7

8.Write a function called swap() that interchanges two int values passed to it by the call-

ing program. (Note that this function swaps the values of the variables in the calling pro-

gram, not those in the function.) You’ll need to decide how to pass the arguments. Create

a main() program to exercise the function.

ans:

//start No.8

#include <iostream>

using namespace std;

void swap(int&, int&);

int main()

{

int a,b;

cout<< "Enter first number: ";

cin>> a;

cout<< "Enter second number: ";

cin>> b;

//swap the value and make pass by ref

swap(b, a);

cout<< "swap a to b and b to a.\n" << endl;

cout<< "First number: " << a;

cout<< "\nSecond number: " << b << endl;

return 0;

}

void swap(int &b, int &a){

int c = 0;

c = a;

a = b;

b = c;

}

//end No.8

9.Repeat Exercise 8, but instead of two int variables, have the swap() function inter-

change two struct time values (see Exercise 6).

ans:

//start No.9

#include <iostream>

using namespace std;

struct Time{

int hr;

int min;

int sec;

};

void swap(Time&, Time&);

int main()

{

Time t\_1, t\_2;

char chara;

cout<<"Enter first time (format: h-m-s): ";

cin>> t\_1.hr >> chara >> t\_1.min >> chara >> t\_1.sec;

cout<< t\_1.hr << chara << t\_1.min << chara << t\_1.sec << endl;

cout<<"Enter second time (format: h-m-s): ";

cin>> t\_2.hr >> chara >> t\_2.min >> chara >> t\_2.sec;

cout<< t\_2.hr << chara << t\_2.min << chara << t\_2.sec << endl;

//swap

swap(t\_1, t\_2);

cout<< "After swap" << endl;

cout<< "First time after swap: " << t\_1.hr << chara << t\_1.min << chara << t\_1.sec << endl;

cout<< "Second time after swap: " << t\_2.hr << chara << t\_2.min << chara << t\_2.sec << endl;

}

void swap(Time &t1, Time &t2){

int hr = 0, min=0, sec=0;

hr = t1.hr;

min = t1.min;

sec = t1.sec;

//hour

t1.hr = t2.hr;

t2.hr = hr;

//minutes

t1.min = t2.min;

t2.min = min;

//sec

t1.sec = t2.sec;

t2.sec = sec;

}

//end No.9

10.Write a function that, when you call it, displays a message telling how many times it has

been called: “I have been called 3 times”, for instance. Write a main() program that calls

this function at least 10 times. Try implementing this function in two different ways.

First, use a global variable to store the count. Second, use a local static variable. Which

is more appropriate? Why can’t you use a local variable?

ans:

//global variable

//start No.10

#include <iostream>

using namespace std;

int count = 0;

void coUnt(){

count++;

}

int main() {

// call coUnt() function ten times

for(int i = 0; i < 10; i++) coUnt();

cout<< "I've been called " << count << " times";

return 0;

}

//end No.10 with global variable

//with static keyword variable

#include <iostream>

using namespace std;

int fun\_count(){

static int count = 0;

count++;

return count;

}

int main() {

int num\_of\_time;

for(int i = 0 ; i < 5 ; i++) fun\_count();

cout<< "function has called " << fun\_count() << " times.";

return 0;

}

//end No.10 Static method

12.Revise the four-function fraction calculator from Exercise 12, Chapter 4, so that it uses

functions for each of the four arithmetic operations. They can be called fadd() , fsub() ,

fmul() , and fdiv() . Each of these functions should take two arguments of type struct

fraction , and return an argument of the same type.

ans:

//start No.12

#include <iostream>

using namespace std;

struct fraction

{

int x,y;

};

//add

fraction fadd(fraction f1, fraction f2)

{

fraction result;

result.x = f1.x\*f2.y + f2.x\*f1.y;

result.y = f1.y\*f2.y;

return result;

}

//substract

fraction fsub(fraction f1, fraction f2)

{

fraction sub\_result;

sub\_result.x = f1.x\*f2.y - f2.x\*f1.y;

sub\_result.y = f1.y\*f2.y;

return sub\_result;

}

//multiply

fraction fmulti(fraction f1, fraction f2)

{

fraction mul\_result;

mul\_result.x = f1.x \* f2.x;

mul\_result.y = f1.y\*f2.y;

return mul\_result;

}

//divide

fraction fdiv(fraction f1, fraction f2)

{

fraction div\_result;

div\_result.x = f1.x\*f2.y + f2.x\*f1.y;

div\_result.y = f1.y\*f2.y;

return div\_result;

}

int main() {

//test

fraction first, sec, result;

char sign, operat;

cout<< "Enter full fraction of operation: (eg. 1/2+2/4): ";

cin>> first.x >> sign >> first.y >> operat >> sec.x >> sign >> sec.y;

switch(operat)

{

case '+':

result = fadd(first, sec);

cout<< result.x << sign << result.y;

break;

case '-':

result = fsub(first, sec);

cout<< result.x << sign << result.y;

break;

case '/':

result = fdiv(first, sec);

cout<< result.x << sign << result.y;

break;

case '\*':

result = fmulti(first, sec);

cout<< result.x << sign << result.y;

break;

default:

cout<< "Enter valid operator for function calling and calculation. Thanks!";

}

result = {0,0};

return 0;

}

//end No.12