**Austin Tesch**

**Chapter Six**

**Date: 05-23-24**

**Program 1: ch6\_pgm1; A program that demonstrates the function “fallingDistance” by calling it in a loop that passes the values 1 through 10 as arguments and displays the return value.**

**Code:**

/\*

Programmer: Austin Tesch

Date modified: 05-23-24

Compiler used:  XCODE v. 15.0

Purpose:

Write a function named "fallingDistance" that accepts an object’s falling time (in seconds) as an argument.

The function should return the distance, in meters, that the object has fallen during that time interval.

Write a program that demonstrates the function by calling it in a loop that passes the values 1 through 10 as arguments and displays the return value.

The following formula can be used to determine the distance an object falls due to gravity in a specific time period:

d=0.5\*g\*t\*t

The variables in the formula are as follows:

d is the distance in meters,

g is 9.8,

and t is the time in seconds that the object has been falling.

\*/

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

double fallingDistance(double fall\_time);

double const GRAVITY = 9.8;

int main(){

// table header

cout << "Fall time" << setw(20) << "Fall Distance" << endl;

// loop through values 1 -10

for (int i = 1; i <= 10; ++i){

// determine distance fallen

double distance = fallingDistance(i);

// output time and distance to table

cout << setw(5) << i << setw(20) << distance << endl;

}

}

double fallingDistance(double fall\_time){

double distance = NULL;

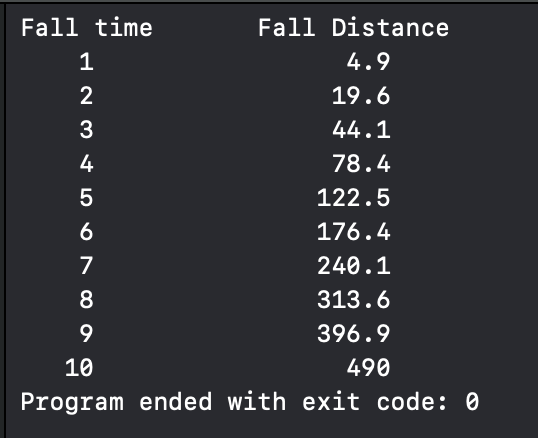
// calculate falling distance in meters

distance = 0.5 \* GRAVITY \* pow(fall\_time, 2);

return distance;

}

**Output:**

****

**Program 2: ch6\_pgm2; A program that calculates the most fuel efficient car over a distance of 500 miles.**

**Code:**

/\*

Programmer: Austin Tesch

Date modified: 05-23-24

Compiler used:  XCODE v. 15.0

Purpose:

Three cars drive a 500-mile route. Write a program that inputs the car's make and the number of gallons of fuel used by each car.

After calling a "calcMPG()" function once for each car, have main determine and display which car was the most fuel-efficient and how many miles per gallon it got.

The "calcMPG()" function should be passed the distance driven and the gallons of gas used as arguments, and should return the miles per gallon obtained.

\*/

#include <iostream>

#include <string>

using namespace std;

double calcMPG(double distance, double gallons);

void inputRequest(string& make, double& gallons, int car\_number);

const int DISTANCE = 500;

const int TOTAL\_CARS = 3;

int main() {

string make1, make2, make3;

double gallons1 = 0.0, gallons2 = 0.0, gallons3 = 0.0; // initlize variables to clear uninitialized variable warning

// Request input from user

for (int i = 1; i <= TOTAL\_CARS; ++i) {

// Store make and gallons of fuel

switch (i) {

case 1:

inputRequest(make1, gallons1, i);

break;

case 2:

inputRequest(make2, gallons2, i);

break;

case 3:

inputRequest(make3, gallons3, i);

break;

}

}

// Calculate MPG

double mpg1 = calcMPG(DISTANCE, gallons1);

double mpg2 = calcMPG(DISTANCE, gallons2);

double mpg3 = calcMPG(DISTANCE, gallons3);

// Determine the most fuel-efficient make

double mpg\_max = mpg1;

string most\_efficient = make1;

if (mpg2 > mpg\_max) {

mpg\_max = mpg2;

most\_efficient = make2;

}

if (mpg3 > mpg\_max) {

mpg\_max = mpg3;

most\_efficient = make3;

}

// Output results

for (int i = 1; i <= TOTAL\_CARS; ++i) {

switch (i) {

case 1:

cout << "The " << make1 << " achieved " << mpg1 << " miles per gallon." << endl;

break;

case 2:

cout << "The " << make2 << " achieved " << mpg2 << " miles per gallon." << endl;

break;

case 3:

cout << "The " << make3 << " achieved " << mpg3 << " miles per gallon." << endl;

break;

}

}

cout << "\nThe most efficient car is the " << most\_efficient << " getting " << mpg\_max << " miles per gallon." << endl;

return 0;

}

double calcMPG(double distance, double gallons) {

return distance / gallons;

}

void inputRequest(string& make, double& gallons, int car\_number) {

cout << "Provide the make of car " << car\_number << ": " << endl;

cin >> make;

cout << "Provide the total amount of fuel used (in gallons) to travel " << DISTANCE << ": " << endl;

cin >> gallons;

}

**Output:**

**A screenshot of a computer

Description automatically generated**

**Program 3: ch6\_pgm3; A program that prompts the user to enter the account’s present value, monthly interest rate, and the number of months that the money will be left in the account and displays the future value.**

**Code:**

/\*

Programmer: Austin Tesch

Date modified: 05-23-24

Compiler used:  XCODE v. 15.0

Purpose:

A program that prompts the user to enter the account’s present value, monthly interest rate, and the number of months that the money will be left in the account.

The program should pass these values to a function named "futureValue" that computes and returns the future value of the account after the specified number of months.

The program should display the account’s future value.

-----------------------------------------------------------------------

Suppose you have a certain amount of money in a savings account that earns compound monthly interest, and you want to calculate the amount that you will have after a specific number of months. The formula, which is known as the future value formula, is:

F=P\*pow(1+i, t)

The variables in the formula are as follows:

F is the future value of the account after the specified time period.

P is the present value of the account.

i is the monthly interest rate.

t is the number of months.

\*/

#include <iostream>

#include <iomanip>

using namespace std;

void grabInput(double& starting\_value, double& intrest\_rate, double& time\_frame);

double calcFutureValue(double starting\_value, double intrest, double time);

int main(){

// get user input

double starting\_value, intrest\_rate, time\_frame;

grabInput(starting\_value, intrest\_rate, time\_frame);

// get future value

double future\_value = calcFutureValue(starting\_value, intrest\_rate, time\_frame);

// display future value

cout << "In " << time\_frame << " months this account will have a value of $" << setprecision(2) << fixed << future\_value << endl;

return 0;

}

void grabInput(double& starting\_value, double& intrest\_rate, double& time\_frame){

// request accounts present value

cout << "What is the present value in this account? " << endl;

cin >> starting\_value;

// request monthly intrest rate

cout << "What is the monthly intrest rate on this account (as a decimal)? " << endl;

cin >> intrest\_rate;

// request number of months money will sit

cout << "How long do you expect to let this money sit in this account (in months)? " << endl;

cin >> time\_frame;

}

double calcFutureValue(double starting\_value, double intrest, double time){

// F = P \* pow(1 + i, t)

double future\_value = starting\_value \* pow(1 + intrest, time);

return future\_value;

}

**Output:**

**A screenshot of a computer

Description automatically generated**

**Program 4: ch6\_pgm4; A program that computes and displays the charges for a patient’s hospital stay.**

**Code:**

/\*

Programmer: Austin Tesch

Date modified: 05-23-24

Compiler used:  XCODE v. 15.0

Purpose:

A program that computes and displays the charges for a patient’s hospital stay.

First, the program should ask if the patient was admitted as an inpatient or an outpatient.

If the patient was an inpatient, the following data should be entered:

The number of days spent in the hospital

The daily rate

Charges for hospital services (lab tests, etc.)

Hospital medication charges

If the patient was an outpatient, the following data should be entered:

Charges for hospital services (lab tests, etc.)

Hospital medication charges

Use a single, separate function to validate that no input is less than zero. If it is, it should be reentered before being returned.

Once the required data has been input and validated, the program should use two overloaded functions to calculate the total charges.

One of the functions should accept arguments for the inpatient data, while the other function accepts arguments for outpatient data. Both functions should return the total charges.

\*/

#include <iostream>

#include <string>

using namespace std;

int validateInput(string prompt);

double calcCost(double days, double rate, double extras, double meds);

double calcCost(double charges, double meds);

int main(){

// determine if patient is inpatient or outpatient

string patient\_status;

double total\_charges;

cout << "Is the patient and inpatient (i) or and outpatient (o)?" << endl;

cin >> patient\_status;

while (true){

// collect charges for inpatient

if (patient\_status == "i" || patient\_status == "I" || patient\_status == "inpatient"){

double total\_days = validateInput("Input total lenght of patient stay (in days): ");

double day\_rate = validateInput("Input the daily rate: ");

double extra\_charges = validateInput("Input total charges for extra hospital services (lab tests, etc.): ");

double med\_charges = validateInput("Input total cost for medications: ");

// calculate total cost

total\_charges = calcCost(total\_days, day\_rate, extra\_charges, med\_charges);

// visual buffer

cout << "-------------------------------- \n" << endl;

// display total cost

cout << "Total charges for this visit are: $" << total\_charges << endl;

break;

}

else if (patient\_status == "o" || patient\_status == "O" || patient\_status == "outpatient"){

// collect charges for outpatient

double hospital\_services = validateInput("Input total cost for hospital services: ");

double med\_charges = validateInput("Input total cost for medications: ");

// calculate total cost

total\_charges = calcCost(hospital\_services, med\_charges);

// visual buffer

cout << "-------------------------------- \n" << endl;

// display total cost

cout << "Total charges for this visit are: $" << total\_charges << endl;

break;

}

else{

// invalid patient status

cout << "Please input i or inpatient for inpatients." << endl;

cout << "Please input o or outpatient for outpatients." << endl;

cin >> patient\_status;

}

}

return 0;

}

int validateInput(string prompt){

// validate that input is greater than zero

double value;

while (true){

cout << prompt;

cin >> value;

// input greater than zero, return

if (value > 0){

return value;

}

else {

// input less than zero, request new input

cout << "Value should be greater than zero." << endl;

}

}

}

double calcCost(double days, double rate, double extras, double meds){

// calculate total cost for inpatient charges

double total\_cost = (days \* rate) + extras + meds;

return total\_cost;

}

double calcCost(double charges, double meds){

// calculate total cost for outpatient charges

double total\_cost = charges + meds;

return total\_cost;

}

**Output:**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**