Chapter 7 Programs

Austin Tesch

Date: 05-31-24

# Program 1: ch7\_pgm1; A program that asks the user for the number of widgets that have been ordered and then displays the number of days it will take to produce them.

## Code:

## main.cpp:

/\*

Programmer: Austin Tesch

Date modified: 05-30-24

Compiler used:  XCODE v. 15.0

Purpose:

A program that asks the user for the number of widgets that have been ordered and then displays the number of days it will take to produce them.

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

Think about what values your program should accept for the number of widgets ordered.

Design a class for a widget manufacturing plant.

Assuming that 10 widgets may be produced each hour, the class object will calculate how many days it will take to produce any number of widgets. (The plant operates two 8-hour shifts per day.)

\*/

#include <iostream>

#include "WidgetProduction.h"

using namespace std;

int main(){

WidgetProduction user\_order;

int widgets\_ordered;

cout << "How many widgets did you order?: " << endl;

cin >> widgets\_ordered;

cout << "This order will take " << user\_order.calcProductionTime(widgets\_ordered) << " days to process." << endl;

return 0;

}

## widgetProduction.h:

//

// WidgetProduction.h

// ch7\_pmg1

//

// Created by Austin Tesch on 5/31/24.

//

#ifndef WidgetProduction\_h

#define WidgetProduction\_h

class WidgetProduction{

private:

double const HOURLY\_PRODUCTION = 10, SHIFT\_LENGTH = 8, DAILY\_SHIFTS = 2;

int widgets;

public:

double calcProductionTime(int);

};

#endif /\* WidgetProduction\_h \*/

## widget\_production.cpp:

//

// widget\_production.cpp

// ch7\_pmg1

//

// Created by Austin Tesch on 5/31/24.

//

#include "WidgetProduction.h"

double WidgetProduction::calcProductionTime(int widgets){

// calculate how many days to produce any number of widgets

// (total\_widgets / 10) - total hours needed

// (total\_hours\_needed / 8) - total shifts needed

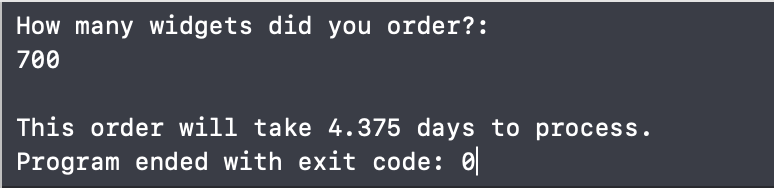
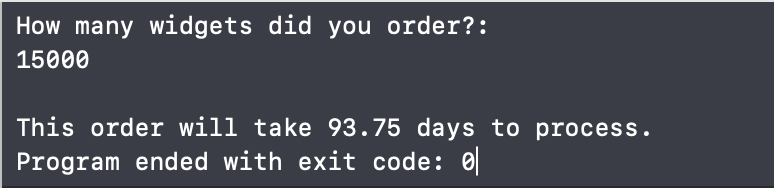
// (total\_shifts / 2) - total days needed

double time\_frame = ((widgets / HOURLY\_PRODUCTION) / SHIFT\_LENGTH) / DAILY\_SHIFTS;

return time\_frame;

}

# Output:



# Program 2: ch7\_pmg2; A program that uses a structure named CorpData to store the following information on a company division: division name (e.g., East, West, North, or South), qtr1 sales, qtr2 sales, qtr3 sales, and qtr4 sales.

## Code:

/\*

Programmer: Austin Tesch

Date modified: 05-31-24

Compiler used:  XCODE v. 15.0

Purpose:

A program that uses a structure named CorpData to store the following information on a company division:

division name (e.g., East, West, North, or South), qtr1 sales, qtr2 sales, qtr3 sales, and qtr4 sales.

Include a constructor that allows the division name and four quarterly sales amount to be specified at the time a CorpData variable is created.

The program should create four CorpData variables, each representing one of the following corporate divisions:

East, West, North, and South.

These variables should be passed one at a time, as constant references, to a function that computes the division’s annual sales total and quarterly average, and displays these along with the division name.

\*/

#include <iostream>

#include <string>

using namespace std;

struct CorpData{

string division;

double q1\_sales, q2\_sales, q3\_sales, q4\_sales;

// constructor

CorpData(string location, double q1, double q2, double q3, double q4) :

division(location), q1\_sales(q1), q2\_sales(q2), q3\_sales(q3), q4\_sales(q4) {}

};

void CalcDivisionData(const CorpData& division);

int main(){

CorpData east("East", 10000.0, 12000.0, 11000.0, 13000.0);

CorpData west("West", 9000.0, 15000.0, 14000.0, 16000.0);

CorpData north("North", 20000.0, 22000.0, 21000.0, 23000.0);

CorpData south("South", 18000.0, 17000.0, 19000.0, 16000.0);

CalcDivisionData(east);

CalcDivisionData(west);

CalcDivisionData(north);

CalcDivisionData(south);

return 0;

}

void CalcDivisionData(const CorpData& division){

// calculate annual sales and quarterly sales

double annual\_sales = division.q1\_sales + division.q2\_sales + division.q3\_sales + division.q4\_sales;

double avarage\_quarterly\_sales = annual\_sales / 4;

// sales information output

cout << "DIVISION: " << division.division << endl;

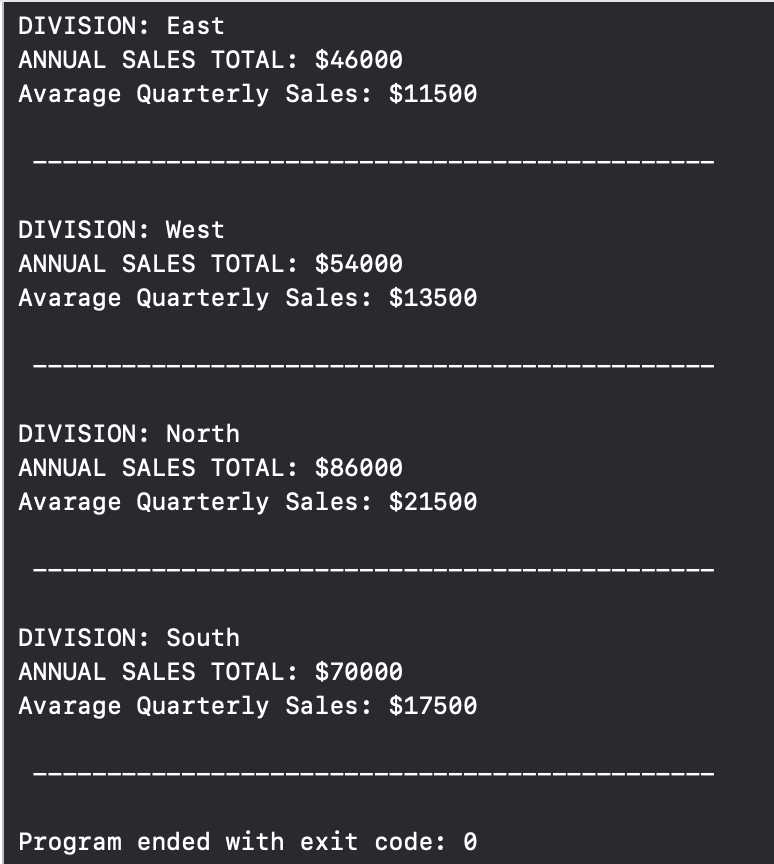
cout << "ANNUAL SALES TOTAL: $" << annual\_sales << endl;

cout << "Avarage Quarterly Sales: $" << avarage\_quarterly\_sales << endl;

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

}

## Output:



# Program 3: ch7\_pgm3; A program to demonstrate the class by creating a menu-driven program that creates a teamScore object and then includes a loop to display a menu and call the appropriate class function depending on the user-entered menu choice.

## Code:

### main.cpp:

/\*

Programmer: Austin Tesch

Date modified: 05-31-24

Compiler used:  XCODE v. 15.0

Purpose:

Youth soccer teams earn 3 points for each win, 1 point for each tie, and 0 points for each loss.

Create a teamScore class with variables to hold the teamName and count the number of wins, ties, and losses.

The constructor should accept the team name and initialize the three counters to 0.

The class should have member functions updateWins, updateTies, and updateLosses that each add 1 to the appropriate counter.

It should also have a displayRecord function that produces a nicely formatted display containing the team name, the number of wins, ties, and losses so far, and the computed points.

Demonstrate the class by creating a menu-driven program that creates a teamScore object and then includes a loop to display a menu and call the appropriate class function depending on the user-entered menu choice.

\*/

#include <iostream>

#include <string>

#include <iomanip>

#include "teamScore.hpp"

#include "menuOperations.hpp"

using namespace std;

int main(){

// init menu

menuOperations menu;

// get team from user

menu.getTeam();

// start menu

menu.startMenu();

return 0;

}

### teamScore.h:

//

// teamScore.hpp

// ch7\_pgm3

//

// Created by Austin Tesch on 5/31/24.

//

#ifndef teamScore\_hpp

#define teamScore\_hpp

#include <string>

using namespace std;

class teamScore{

string team\_name;

int num\_wins, num\_ties, num\_losses;

public:

// constructor

teamScore(const string& team) :

team\_name(team), num\_wins(0), num\_ties(0), num\_losses(0) {}

void updateWins(){

// increment wins by 1

num\_wins++;

}

void updateTies(){

// increment ties by 1

num\_ties++;

}

void updateLosses(){

// increment losses by 1

num\_losses++;

}

// init

void displayRecord(teamScore& team\_name);

};

#endif /\* teamScore\_hpp \*/

### menuOperations.h:

//

// menuOperations.hpp

// ch7\_pgm3

//

// Created by Austin Tesch on 5/31/24.

//

#ifndef menuOperations\_hpp

#define menuOperations\_hpp

#include <iostream>

using namespace std;

class menuOperations{

string team\_name;

public:

void getTeam(){

// get team name from user

cout << "Enter team name: " << endl;

cin >> team\_name;

}

void displayMenu(string team\_name){

// display menu to user

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

cout << "Team: " << team\_name << "\n" << endl;

cout << "1. Update Wins (add 1 win)" << endl;

cout << "2. Update Ties (add 1 tie)" << endl;

cout << "3. Update Losses (add one loss)" << endl;

cout << "4. Display Team Record" << endl;

cout << "5. Pick New Team" << endl;

cout << "6. Exit program" << endl;

}

// init

void startMenu();

};

#endif /\* menuOperations\_hpp \*/

### teamScore.cpp:

//

// teamScore.cpp

// ch7\_pgm3

//

// Created by Austin Tesch on 5/31/24.

//

#include <iostream>

#include <curses.h>

#include "teamScore.hpp"

int const WIN = 3, TIE = 1;

void teamScore::displayRecord(teamScore& team\_name){

// cacl total points for team

int total\_points = (num\_wins \* WIN) + (num\_ties \* TIE);

// display record to user

cout << "Team: " << team\_name.team\_name << endl;

cout << "Total wins: " << team\_name.num\_wins << endl;

cout << "Total Ties: " << team\_name.num\_ties << endl;

cout << "Total losses: " << team\_name.num\_losses << endl;

cout << "Total points: " << total\_points << endl;

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

}

teamScore.cpp:

//

// menuOperations.cpp

// ch7\_pgm3

//

// Created by Austin Tesch on 5/31/24.

//

#include <iostream>

#include "menuOperations.hpp"

#include "teamScore.hpp"

menuOperations menu;

void menuOperations::startMenu(){

teamScore team(team\_name);

int user\_choice = NULL;

do {

// display menu and watch for input

displayMenu(team\_name);

cin >> user\_choice;

switch (user\_choice) {

case 1:

// update wins by 1

team.updateWins();

break;

case 2:

// update ties by 1

team.updateTies();

break;

case 3:

//update losses by 1

team.updateLosses();

break;

case 4:

// display team record

team.displayRecord(team);

break;

case 5:

// get new team

menu.getTeam();

// start menu

menu.startMenu();

break;

case 6:

// exit program

cout << "Exiting." << endl;

break;

default:

// invalid input

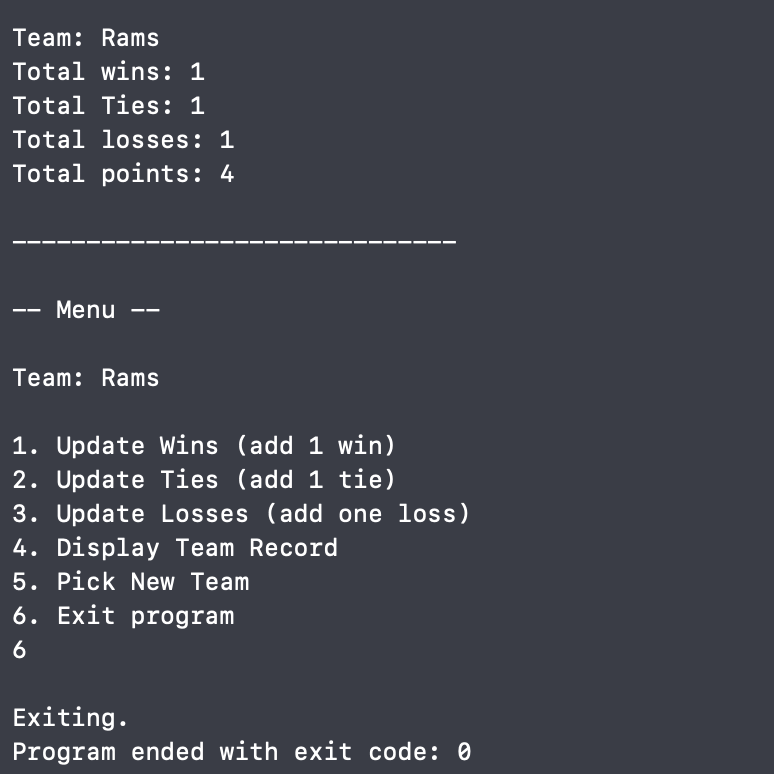
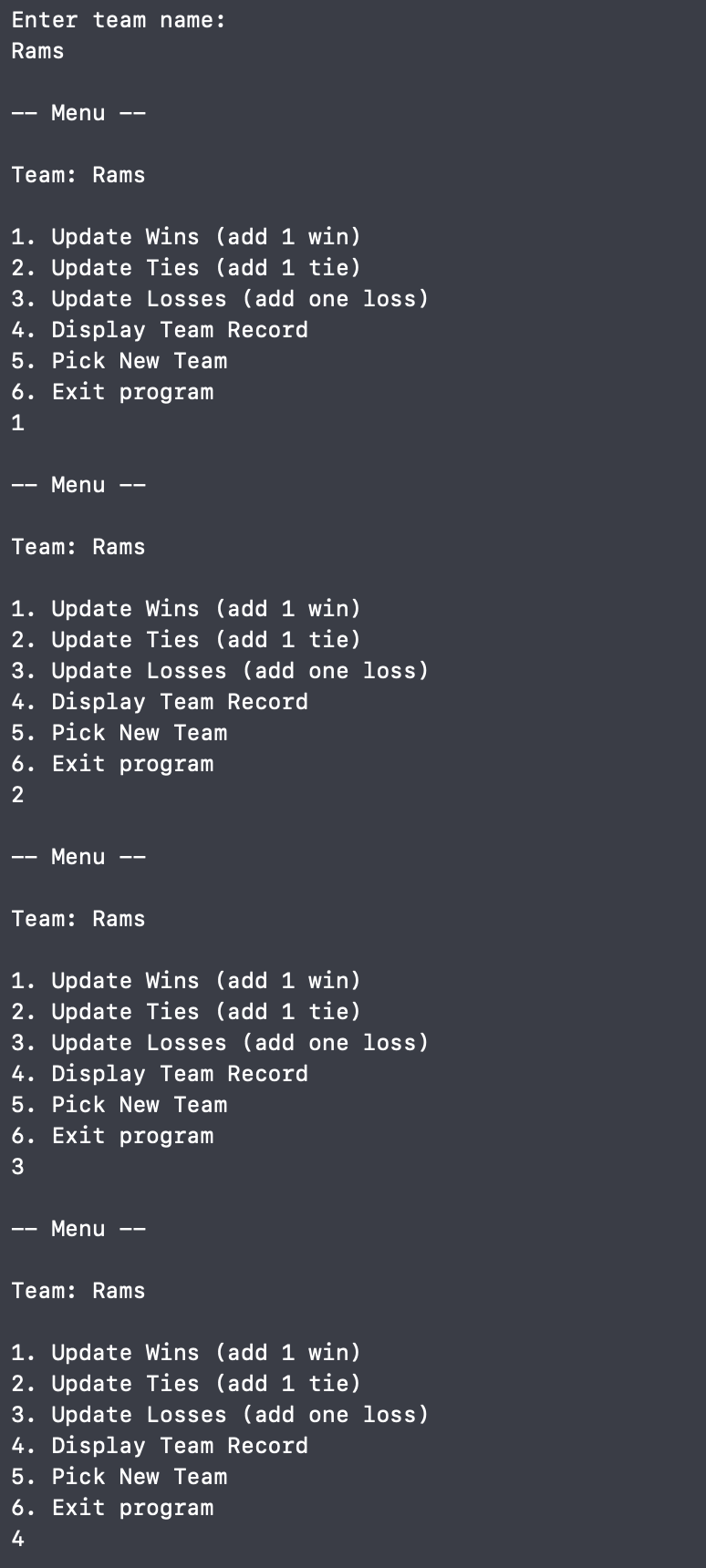
cout << "Invalid input." << endl;

}

} while (user\_choice != 5);

}

## Output:



# Program 4: ch7\_pgm4; a program that demonstrates the Coin class. The program should create an instance of the class and display the side that is initially facing up.

## Code:

/\*

Programmer: Austin Tesch

Date modified: 05-32-24

Compiler used:  XCODE v. 15.0

Purpose:

Write a program that demonstrates the Coin class. The program should create an instance of the class and display the side that is initially facing up.

Then, use a loop to toss the coin 20 times. Each time the coin is tossed, display the side that is facing up.

The program should keep count of the number of times heads is facing up and the number of times tails is facing up, and display those values after the loop finishes.

Write a class named Coin. The Coin class should have the following member variable:

A string named sideUp. The sideUp member variable will hold either “heads” or “tails” indicating the side of the coin that is facing up.

The Coin class should have the following member functions:

A default constructor that randomly determines the side of the coin that is facing up (“heads” or “tails”) and initializes the sideUp member variable accordingly.

A void member function named "toss" simulates the tossing of the coin. When the toss member function is called, it randomly determines the side of the coin that is facing up (“heads” or “tails”) and sets the sideUp member variable accordingly.

A member function named getSideUp returns the value of the sideUp member variable.

\*/

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <string>

using namespace std;

class Coin {

private:

string sideUp;

public:

// Default constructor that randomly sets the sideUp variable

Coin() {

toss();

}

// simulate tossing the coin

void toss() {

int randomValue = rand() % 2; // Generate 0 or 1

if (randomValue == 0) {

sideUp = "heads";

} else {

sideUp = "tails";

}

}

// return the value of sideUp

string getSideUp() {

return sideUp;

}

};

int main() {

srand(static\_cast<unsigned int>(time(0))); // Seed the random number generator

Coin myCoin;

int headsCount = 0;

int tailsCount = 0;

// Display the initial side facing up

cout << "Initial side up: " << myCoin.getSideUp() << endl;

// Toss the coin 20 times

for (int i = 0; i < 20; ++i) {

myCoin.toss();

string result = myCoin.getSideUp();

cout << "Toss " << i + 1 << ": " << result << endl;

// Update the counts

if (result == "heads") {

++headsCount;

} else {

++tailsCount;

}

}

// Display the final counts

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

cout << "Final counts: " << endl;

cout << "Heads count: " << headsCount << endl;

cout << "Tails count: " << tailsCount << endl;

return 0;

}

## Output:

