# Group 3: Nguyễn Đức Duy Nguyễn Công Hoàng Trần Quốc Bảo

Project 4 (Chapter 6,7)

## Checkpoint

#### 1. The following program skeleton determines whether a person qualifies for a credit card. To qualify, the person must have worked on his or her current job for at least two years and make at least $17,000 per year. Finish the program by writing the definitions of the functions qualify and noQualify . The function qualify should explain that the applicant qualifies for the card and that the annual interest rate is 12%. The function noQualify should explain that the applicant does not qualify for the card and give a general explanation why. (6.4)

#include <iostream>

**using namespace** std;

**void** qualify() {

cout << "You are qualified for a credit card.\n"

<< "Your annual interest rate is 12%.";

}

**void** noQualify() {

cout << "You are not qualified for a credit card.\n"

<< "In order to be qualified you need:\n"

<< "\t+ Annual Income at least $17,000.00\n"

<< "\t+ Continuously working on the current job for at least the last 2 years.\n";

}

**int** main() {

**double** salary;

**int** years;

cout << "This program will determine if you qualify\n";

cout << "for our credit card.\n";

cout << "What is your annual salary? ";

cin >> salary;

cout << "How many years have you worked at your ";

cout << "current job? ";

cin >> years;

**if** (salary >= 17000.0 && years >= 2)

qualify();

**else**

noQualify();

**return** 0;

}

#### 2. Write a function named timesTen The function should have an integer parameter named number . When timesTen is called, it should display the product of number times ten. (Note: just write the function. Do not write a complete program.) (6.6)

**void** timesTen(**int** number) {

cout << number\*10 << endl;

}

#### 3.The following program skeleton asks for the number of hours you’ve worked and your hourly pay rate. It then calculates and displays your wages. The function showDollars , which you are to write, formats the output of the wages. (6.10)

#include <iostream>  
#include <iomanip>

**using namespace** std;

**void** showDollars(**double**); // Function prototype

**int** main()

{

**double** payRate, hoursWorked, wages;

cout << "How many hours have you worked? ";

cin >> hoursWorked;

cout << "What is your hourly pay rate? ";

cin >> payRate;

wages = hoursWorked \* payRate;

showDollars(wages);

**return** 0;

}

// You must write the definition of the function showDollars

// here. It should take one parameter of the type double.

// The function should display the message "Your wages are $"

// followed by the value of the parameter. It should be displayed

// with 2 places of precision after the decimal point, in fixed

// notation, and the decimal point should always display.

**void** showDollars(**double** wage) {

cout << fixed << showpoint << setprecision(2);

cout << "Your wages are $" << wage;

}

## Programming Challenges

#### 1. Markup Write a program that asks the user to enter an item’s wholesale cost and its markup percentage. It should then display the item’s retail price. For example: - If an item’s wholesale cost is 5.00 and its markup percentage is 100%, then the item’s retail price is 10.00.

#### - If an item’s wholesale cost is 5.00 and its markup percentage is 50%, then the item’s retail price is 7.50. The program should have a function named calculateRetail that receives the wholesale cost and the markup percentage as arguments, and returns the retail price of the item. Input Validation: Do not accept negative values for either the wholesale cost of the item or the markup percentage. (6.1)

#include <iostream>

#include <iomanip>

**using namespace** std;

**double** calculateRetail(**double**, **double**);

**int** main() {

**double** cost, markup100;

cout << fixed << showpoint << setprecision(2);

cout << "Item wholesale cost:\t";

**do** {

cin >> cost;

**if** (cost < 0)

cout << "Wholesale cost cannot be negative. Enter again:\t";

} **while** (cost < 0);

cout << "Item markup percentage (20% -> 20):\t";

**do** {

cin >> markup100;

**if** (markup100 < 0)

cout << "Markup percentage cannot be negative. Enter again:\t";

} **while** (markup100 < 0);

cout << "\n\nRetail price of the item:\t$"

<< calculateRetail(cost, markup100 / 100);

**return** 0;

}

**double** calculateRetail(**double** cost, **double** markup) {

**return** cost \* (1 + markup);

}

#### 2. Rectangle Area—Complete the Program The Student CD contains a partially written program named AreaRectangle.cpp . Your job is to complete the program. When it is complete, the program will ask the user to enter the width and length of a rectangle, and then display the rectangle’s area.

#### The program calls the following functions, which have not been written: - getLength – This function should ask the user to enter the rectangle's length, and then return that value as a double . - getWidth - This function should ask the user to enter the rectangle's width, and then return that value as a double . - getArea – This function should accept the rectangle's length and width as arguments, and return the rectangle's area. The area is calculated by multiplying the length by the width. - displayData – This function should accept the rectangle's length, width, and area as arguments, and display them in an appropriate message on the screen. (6.2)

#include <iostream>  
#include <iomanip>

**using namespace** std;

**double** getLength();

**double** getWidth();

**double** getArea(**double**, **double**);

**void** displayData(**double**, **double**, **double**);

**int** main() {

**double** l = getLength(), w = getWidth();

cout << "\n\n";

displayData(l, w, getArea(l, w));

**return** 0;

}

**double** getLength() {

**double** l;

cout << "Enter rectangle length:\t";

**do** {

cin >> l;

**if** (l <= 0)

cout << "Length must > 0. Enter again:\t";

} **while** (l <= 0);

**return** l;

}

**double** getWidth() {

**double** w;

cout << "Enter rectangle width:\t";

**do** {

cin >> w;

**if** (w <= 0)

cout << "Width must > 0. Enter again:\t";

} **while** (w <= 0);

**return** w;

}

**double** getArea(**double** width, **double** height) {

**return** width \* height;

}

**void** displayData(**double** length, **double** width, **double** area) {

**const int** OFFSET = 20;

**int** m\_l = **static\_cast**<**int**> (length);

**int** m\_w = **static\_cast**<**int**> (width);

**bool** b = **false**;

m\_l = m\_l > 14 ? m\_l : 14;

m\_w = m\_w > 5 ? m\_w : 5;

cout << setw(OFFSET) << "" << setw(m\_l) << "Length: " << length << endl;

**for** (**int** i = 0; i < m\_w; i++) {

**if** (i == (m\_w - 1) / 2)

cout << setw(OFFSET - 3 - 6) << "Width: " << setw(6) << left << width << " ";

**else**

cout << setw(OFFSET) << "";

**for** (**int** j = 0; j < m\_l; j++) {

**if** (i == 0 || i == m\_w - 1 || j == 0 || j == m\_l - 1)

cout << "\* ";

**else if** (i == (m\_w - 1) / 2) {

**if** (!b)

cout << setw(m\_l - 2) << right<< " Area: "

<< setw(6) << left << area

<< setw((m\_l - 2) - 6) << "";

b = **true**;

} **else**

cout << " ";

}

cout << "\n";

}

}

**2nd version:**

#include <iostream>  
#include <iomanip>

**using namespace** std;

**double** getLength(){

**double** length;

cout << "Enter the length of the rectangle: ";

cin >> length;

**return** length;

}

**double** getWidth(){

**double** width;

cout << "Enter the width of the rectangle: ";

cin >> width;

**return** width;

}

**double** getArea(**double** length, **double** width){

**return** length \* width;

}

**void** displayData(**double** length, **double** width, **double** area){

cout << "The length of the rectangle is: " << length << endl;

cout << "The width of the rectangle is: " << width << endl;

cout << "The area of the rectangle is: " << area << endl;

}

**int** main(){

**double** length, width, area;

length = getLength();

width = getWidth();

area = getArea(length, width);

cout << endl;

displayData(length, width, area);

}

#### 3.Falling Distance

#### When an object is falling because of gravity, the following formula can be used to determine the distance the object falls in a specific time period:

#### The variables in the formula are as follows: d is the distance in meters, g is 9.8, and t is the amount of time, in seconds, that the object has been falling.

#### 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. (6.5)

#include <iostream>  
#include <iomanip>

**using namespace** std;

**double** fallingDistance(**double**);

**int** main() {

cout << setw(10) << "Time" << setw(15) << "Distance" << endl;

cout << fixed << showpoint << setprecision(2);

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

cout << setw(10) << i << setw(15) << fallingDistance(i + 0.0) << endl;

}

**return** 0;

}

**double** fallingDistance(**double** time) {

**static const double** g = 9.8;

**return** 0.5 \* g \* time \* time;

}

#### 4.Present Value

#### Suppose you want to deposit a certain amount of money into a savings account, and then leave it alone to draw interest for the next 10 years. At the end of 10 years you would like to have $10,000 in the account. How much do you need to deposit today to make that happen? You can use the following formula, which is known as the present value formula, to find out:

#### The terms in the formula are as follows:

#### - P is the present value, or the amount that you need to deposit today.

#### - F is the future value that you want in the account. (In this case, F is $10,000.)

#### - r is the annual interest rate.

#### - n is the number of years that you plan to let the money sit in the account.

#### Write a program that has a function named presentValue that performs this calculation. The function should accept the future value, annual interest rate, and number of years as arguments. It should return the present value, which is the amount that you need to deposit today. Demonstrate the function in a program that lets the user experiment with different values for the formula's terms.

#include <iostream>  
#include <iomanip>

#include <cmath>

**using namespace** std;

**double** presentValue(**double**, **double**, **double**);

**int** main() {

**double** f, t, r100;

cout << "Expected future money:\t$";

cin >> f;

cout << "Expected time:\t";

cin >> t;

cout << "Annual Interest rate: (12% -> 12)\t";

cin >> r100;

cout << "\n\n\n";

cout << "Money need to deposit:\t$"

<< presentValue(f, r100/100, t);

**return** 0;

}

**double** presentValue(**double** futureMoney, **double** interest, **double** years) {

**return** futureMoney / pow(1 + interest, years);

}

#### 5. Rainfall Statistics

#### Write a program that lets the user enter the total rainfall for each of 12 months into an array of double s. The program should calculate and display the total rainfall for the year, the average monthly rainfall, and the months with the highest and lowest amounts.

#### Input Validation: Do not accept negative numbers for monthly rainfall figures.

#include <iostream>

#include <iomanip>

**using namespace** std;

**int** main(){

**double** rain[12], total = 0.0;

**for** (**int** i = 0; i < 12; ++i){

**do**{

cout << "Enter the rain fall of month " << i + 1 << " (mm): ";

cin >> rain[i];

cin.ignore(256, '\n');

**if** (rain[i] < 0)

cout << "Invalid rainfall." << endl;

total += rain[i];

} **while** (rain[i] < 0);

}

**int** i\_highest = 0, i\_lowest = 0;

**double** r\_highest = rain[0], r\_lowest = rain[0];

**for** (**int** i = 1; i < 12; ++i){

**if**(r\_lowest > rain[i]){

i\_lowest = i;

r\_lowest = rain[i];

}

**if**(r\_highest < rain[i+1]){

i\_highest = i;

r\_highest = rain[i];

}

}

cout << fixed << setprecision(2)

<< "The total rainfall for the year is " << total

<< " mm" << endl << "The average monthly rainfall is "

<< total / 12.0 << " mm" << endl

<< "The months with the highest amount is month " << i\_highest + 1

<< " with " << r\_highest << endl

<< "The months with the lowest amount is month "

<< i\_lowest + 1 << " with " << r\_lowest << endl;

**return** 0;

}

#### 6. Chips and Salsa Write a program that lets a maker of chips and salsa keep track of sales for five differ-

#### ent types of salsa: mild, medium, sweet, hot, and zesty. The program should use two parallel 5-element arrays: an array of strings that holds the five salsa names and an array of integers that holds the number of jars sold during the past month for each salsa type. The salsa names should be stored using an initialization list at the time the name array is created. The program should prompt the user to enter the number of jars sold for each type. Once this sales data has been entered, the program should produce a report that displays sales for each salsa type, total sales, and the names of the highest selling and lowest selling products.

#### Input Validation: Do not accept negative values for number of jars sold. (7.3)

#include <iostream>

#include <iomanip>

#include <string>

**using namespace** std;

**int** main() {

string salsaType[5] = {"mild", "medium", "sweet", "hot", "zesty"};

**int** jarSold[5], total = 0;

cout << "Enter the jars sold for each type of salsa: " << endl;

**for** (**int** i = 0; i < 5; ++i) {

cout << "\t" << salsaType[i] << ": ";

cin >> jarSold[i];

cin.ignore(256, '\n');

total += jarSold[i];

}

string highestType = salsaType[0],

lowestType = salsaType[0];

**int** highestJar = jarSold[0],

lowestJar = jarSold[0];

**for** (**int** i = 1; i < 5; ++i) {

**if** (highestJar < jarSold[i]) {

highestJar = jarSold[i];

highestType = salsaType[i];

}

**if** (lowestJar > jarSold[i]) {

lowestJar = jarSold[i];

lowestType = salsaType[i];

}

}

cout << left << setw(10) << "Type" << setw(25)

<< "Numbers of Jars Sold" << endl;

**for** (**int** i = 0; i < 5; ++i) {

cout << setw(10) << salsaType[i] << setw(10)

<< jarSold[i] << endl;

}

cout << setw(10) << "Total" << setw(25) << total << endl;

cout << "The highest selling product is " << highestType

<< " with " << highestJar << " jars" << endl;

cout << "The lowest selling product is " << lowestType

<< " with " << lowestJar << " jars" << endl;

**return** 0;

}

#### 7. Number Analysis Program

#### Write a program that asks the user for a file name. Assume the file contains a series of numbers, each written on a separate line. The program should read the contents of the file into an array and then display the following data:

#### - The lowest number in the array

#### - The highest number in the array

#### - The total of the numbers in the array

#### - The average of the numbers in the array

#### The Student CD contains a text file named numbers.txt . that you can use to test the program. (7.6)

#include <iostream>  
#include <fstream>

#include <iomanip>

**using namespace** std;

**void** countNum(**unsigned**&);

**void** getNum(**unsigned**&);

**int** main(){

**unsigned** count;

countNum(count);

getNum(count);

**return** 0;

}

**void** countNum(**unsigned**& count){

ifstream inputFile;

inputFile.open("numbersFor7.6.txt");

**if**(!inputFile) cout << "Error opening file." << endl;

**double** a;

**while** (inputFile >> a)

count++;

cout << "The file contains " << count << " numbers." << endl;

}

**void** getNum(**unsigned**& count){

ifstream inputFile;

inputFile.open("numbersFor7.6.txt");

**if**(!inputFile) cout << "Error opening file." << endl;

// get numbers

**double** arr[count], total = 0.0;

**for** (**int** i = 0; i < count; ++i){

inputFile >> arr[i];

total += arr[i];

}

// highest, lowest

**double** highest = arr[0], lowest = arr[0];

**for**(**int** i = 1; i < count; ++i){

**if** (lowest > arr[i]){

lowest = arr[i];

}

**if** (highest < arr[i]){

highest = arr[i];

}

}

// display data

**int** j = 0;

**for** (**int** i = 0; i < count; ++i){

cout << arr[i] << " ";

j++;

**if** (j == 10) {

cout << endl;

j = 0;

}

}

cout << endl

<< fixed << setprecision(2)

<< "Total: " << total << endl

<< "Average: " << total / count << endl

<< "Highest: " << highest << endl

<< "Lowest: " << lowest << endl;

}

#### 8. Lowercase to Uppercase Converter

#### Write a program that lets the user enter a string into a character array. The program should then convert all the lowercase letters to uppercase. (If a character is already uppercase, or is not a letter, it should be left alone.) Hint: Consult the ASCII chart in Appendix A. Notice that the lowercase letters are represented by the ASCII codes 97 through 122. If you subtract 32 from any lowercase character’s ASCII code, it will yield the ASCII code of the uppercase equivalent.

**#include <iostream>**

**using namespace** std;

**int** main() {

**char** arr[1000];

cout << "Enter a string: ";

cin >> arr;

**for** (**int** i = 0; i < 1000; ++i) {

**if** (arr[i] != '\0') {

**if** (arr[i] >= 'a' && arr[i] <= 'z') {

arr[i] -= 32;

}

} **else break**;

}

cout << "UPPERCASE string: " << arr;

**return** 0;

}

## Other

#### 1. In mathematics, a perfect number is defined as a positive integer which is the sum of its proper positive divisors, that is, the sum of the positive divisors not including the number it self. Eg: 6 = 1 + 2 + 3

#### Write a program that lists perfect numbers which is smaller than inputed N.

#include <iostream>  
#include <cmath>

**using namespace** std;

**bool** isPerfect(**int** num) {

**int** sum = 0;

**for** (**int** count = 1; count < num; ++count) {

**if** (num % count == 0)

sum += count;

}

**if** (sum == num)

**return true**;

**return false**;

}

**bool** isPrime(**int** num) {

**int** bound;

bound = sqrt(num);

**if** (num == 2) {

**return true**;

} **else** {

**for** (**int** count = 2; count <= bound; ++count) {

**if** ((num % count) == 0) {

**return false**;

**break**;

} **else if** (count == bound)

**return true**;

}

}

}

**int** main() {

**int** choice, n = 10, i, j;

**do** {

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

<< " Menu" << endl

<< "------" << endl

<< "1.Enter a number N (default value = 10)." << endl

<< "2.Print perfect numbers from 1 to N." << endl

<< "3.Print prime numbers from 1 to N." << endl

<< "4.Print all divisors of N." << endl

<< "5.Quit." << endl

<< "------" << endl;

cout << "Enter your choice: ";

cin >> choice;

**switch** (choice) {

**case** 1:

cout << "Enter a number: ";

cin >> n;

**break**;

**case** 2:

cout << "Perfect numbers from 1 to " << n << endl;

j = 0;

**for** (i = 2; i <= n; ++i) {

**if** (isPerfect(i) == **true**) {

cout << i << " ";

j++;

**if** (j == 10) {

cout << endl;

j = 0;

}

}

}

**break**;

**case** 3:

cout << "Prime numbers from 1 to " << n << endl;

j = 0;

**for** (i = 2; i <= n; ++i) {

**if** (isPrime(i) == **true**) {

cout << i << " ";

j++;

**if** (j == 10) {

cout << endl;

j = 0;

}

}

}

**break**;

**case** 4:

cout << "All divisors of " << n << endl;

j = 0;

**for** (i = 1; i <= n; ++i) {

**if** (n % i == 0) {

cout << i << " " << -i << " ";

j += 2;

**if** (j == 10) {

cout << endl;

j = 0;

}

}

}

**break**;

**default**:

cout << "Have a problem with numbers?";

}

} **while** (choice != 5);

**return** 0;

}

## Group Project

#include <iostream>  
#include <iomanip>

**using namespace** std;

**void** getTime(**int** &, **double** &, **double** &);

**void** getAirFare(**double** &);

**void** getCarRent(**double** &);

**void** getPrivateDrive(**double** &);

**void** getMeeting(**double** &);

**void** getParking(**double** &, **int** &, **double** &);

**void** getTaxi(**double** &, **int** &, **double** &);

**void** getHotel(**double** &, **int** &, **double** &);

**void** getMeal(**double** &, **double** &, **double** &, **double** &);

**int** main() {

**int** day;

**double** depart, back,

airFare, carRent, privateDrive, parking, taxi,

meeting, hotel, mealFee, total,

parkingAllowed, taxiAllowed, hotelAllowed, mealAllowed,

parkingExcess, taxiExcess, mealExcess, hotelExcess, totalExcess;

cout << "Times for Business Trip" << endl

<< "-----------------------" << endl;

getTime(day, depart, back);

cout << "\nUnallowable Fees" << endl

<< "-----------------" << endl;

getAirFare(airFare);

getCarRent(carRent);

getPrivateDrive(privateDrive);

getMeeting(meeting);

cout << "\nAllowable Fees" << endl

<< "---------------" << endl;

getParking(parking, day, parkingAllowed);

parkingExcess = parking - parkingAllowed;

getTaxi(taxi, day, taxiAllowed);

taxiExcess = taxi - taxiAllowed;

getHotel(hotel, day, hotelAllowed);

hotelExcess = hotel - hotelAllowed;

getMeal(mealFee, depart, back, mealAllowed);

mealExcess = mealFee - mealAllowed;

total = airFare + carRent + privateDrive + parking + taxi + meeting + hotel + mealFee;

totalExcess = parkingExcess + taxiExcess + mealExcess;

cout << "\nCalculations" << endl

<< "-------------" << endl;

cout << left << setw(20) << "\nDays: " << day << setw(20) << "\nAirFare: " << airFare << setw(20) << "\nCar Rentals"

<< carRent

<< setw(20) << "\nPrivateDrive: " << privateDrive << setw(20) << "\nParking: " << parking << setw(20)

<< "\nTaxi: " << taxi

<< setw(20) << "\nConference: " << meeting << setw(20) << "\nHotel: " << hotel << setw(20) << "\nMeal Fee: "

<< mealFee

<< setw(20) << "\n\nTotal Expense: " << total << setw(20) << "\nTotal Allowed: "

<< parkingAllowed + taxiAllowed + hotelAllowed + mealAllowed;

**if** (totalExcess >= 0) {

cout << left << setw(20) << "\nTotal Excess: " << totalExcess;

} **else if** (totalExcess < 0) {

cout << left << setw(20) << "\nTotal Save: " << -totalExcess;

}

**return** 0;

}

**void** getTime(**int** &day, **double** &depart, **double** &back) {

**do** {

cout << "The total number of days spent on the trip? ";

cin >> day;

**if** (day < 1) {

cout << "Invalid number of days." << endl;

}

} **while** (day < 1);

**do** {

cout << "The time of departure on the first day of the trip (from 5 to 23)? ";

cin >> depart;

**if** (depart < 5 || depart > 22) {

cout << "Invalid time." << endl;

}

} **while** (depart < 5 || depart > 22);

**do** {

cout << "The time of arrival back home on the last day of the trip (from 5 to 23)? ";

cin >> back;

**if** (back < 5 || back > 23) {

cout << "Invalid time." << endl;

}

} **while** (back < 5 || back > 23);

}

**void** getAirFare(**double** &airFare) {

cout << "The amount of any round-trip airfare? ";

cin >> airFare;

}

**void** getCarRent(**double** &carRent) {

cout << "The amount of any car rentals? ";

cin >> carRent;

}

**void** getPrivateDrive(**double** &privateDrive) {

**double** mile;

**do** {

cout << "Miles driven, if a private vehicle was used? ";

cin >> mile;

**if** (mile < 0) {

cout << "Invalid number of miles." << endl;

}

} **while** (mile < 0);

privateDrive = mile \* 0.27;

}

**void** getParking(**double** &parking, **int** &day, **double** &parkingAllowed) {

cout << "Parking fees (allow 6$/day)? ";

cin >> parking;

parkingAllowed = 6.0 \* day;

}

**void** getTaxi(**double** &taxi, **int** &day, **double** &taxiAllowed) {

cout << "Taxi fees, if a taxi was used anytime during the trip (allow 10$/day)? ";

cin >> taxi;

taxiAllowed = 10.0 \* day;

}

**void** getMeeting(**double** &meeting) {

cout << "Conference or seminar registration fees? ";

cin >> meeting;

}

**void** getHotel(**double** &hotel, **int** &day, **double** &hotelAllowed) {

cout << "Hotel expenses (allow 90$/day)? ";

cin >> hotel;

hotelAllowed = 90.0 \* day;

}

**void** getMeal(**double** &mealFee, **double** &depart, **double** &back, **double** &mealAllowed) {

**int** breakfast = 0, lunch = 0, dinner = 0, i = 1;

**double** meal;

**if** (depart < 7) {

breakfast++;

} **else if** (depart >= 7 && depart < 12) {

lunch++;

} **else if** (depart >= 12 && depart < 18) {

dinner++;

}

**if** (back > 8 && back <= 13) {

breakfast++;

} **else if** (back > 13 && back <= 19) {

lunch++;

} **else if** (back > 19) {

dinner++;

}

cout << "You have " << breakfast + lunch + dinner << " allowable meal(s): "

<< breakfast << " breakfast (allow 9$), " << lunch << " lunch (allow 12$), " << dinner

<< " dinner (allow 16$)." << endl;

**while** (i > 0 && i <= breakfast + lunch + dinner) {

cout << "\tThe price of meal " << i << "? ";

cin >> meal;

mealFee += meal;

i++;

}

mealAllowed = 9 \* breakfast + 12 \* lunch + 16 \* dinner;

}