**COP 3331 Week 3 Examples:**

1. Struct Example:

// This program demonstrates the use of structures.

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

struct PayRoll

{

int empNumber; // Employee number

string name; // Employee's name

double hours; // Hours worked

double payRate; // Hourly payRate

double grossPay; // Gross Pay

};

int main()

{

PayRoll employee; // employee is a PayRoll structure.

// Get the employee's number.

cout << "Enter the employee's number: ";

cin >> employee.empNumber;

// Get the employee's name.

cout << "Enter the employee's name: ";

cin.ignore(); // To skip the remaining '\n' character

getline(cin, employee.name);

// Get the hours worked by the employee.

cout << "How many hours did the employee work? ";

cin >> employee.hours;

// Get the employee's hourly pay rate.

cout << "What is the employee's hourly payRate? ";

cin >> employee.payRate;

// Calculate the employee's gross pay.

employee.grossPay = employee.hours \* employee.payRate;

// Display the employee data.

cout << endl << "Here is the employee's payroll data:\n";

cout << "Name: " << employee.name << endl;

cout << "Number: " << employee.empNumber << endl;

cout << "hours worked: " << employee.hours << endl;

cout << "Hourly payRate: " << employee.payRate << endl;

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

cout << "Gross Pay: $" << employee.grossPay << endl;

}

1. Arrays and Vectors of Structs:

// This program uses an array and a vector of structures.

#include <iostream>

#include <array>

#include <iomanip>

#include <vector>

using namespace std;

struct PayInfo

{

int hours; // Hours Worked

double payRate; // Hourly Pay Rate

};

int main()

{

const int NUM\_WORKERS = 3; // Number of workers

array <PayInfo, NUM\_WORKERS> workers; // use array template

int index; // Loop counter

// Get employee pay data.

cout << "Enter the hours worked by " << NUM\_WORKERS

<< " employees and their hourly rates.\n";

for (index = 0; index < workers.size(); index++)

{

// Get the hours worked by an employee.

cout << "Hours worked by employee #" << (index + 1);

cout << ": ";

cin >> workers[index].hours;

// Get the employee's hourly pay rate.

cout << "Hourly pay rate for employee #";

cout << (index + 1) << ": ";

cin >> workers[index].payRate;

cout << endl;

}

// Display each employee's gross pay.

cout << "Here is the gross pay for each employee:\n";

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

for (index = 0; index < NUM\_WORKERS; index++)

{

double gross;

gross = workers[index].hours \* workers[index].payRate;

cout << "Employee #" << (index + 1);

cout << ": $" << gross << endl;

}

//Create a vector for new employees

vector <PayInfo> newWorkers;

PayInfo temp; // create a temp structure

//Now fill it up...

cout << "\nHours worked by new employee: ";

cin >> temp.hours;

cout << "Hourly pay rate for new employee: ";

cin >> temp.payRate;

//... and insert it into vector

newWorkers.push\_back(temp); // Yup, you can pass a structure

cout << "New employee pay: $"

<< newWorkers[0].hours \* newWorkers[0].payRate

<< endl;

}

1. Class example with inline functions  
     
   Header File:

/\* Account.h

This is a modified form of the class definition

shown in Fig. 3.2 (page 79) in the 10th edition text.

This example uses inline member functions. \*/

#include <string> // enable this program to use C++ string data type

using namespace std;

class Account

{

private:

string name; // data member containing account holder's name

public:

// mutator function

void setName(string accountName)

{

name = accountName; // store the account name

}

// accessor function

string getName() const

{

return name; // return name's value to this function's caller

}

};

Driver Program:

/\* Driver program for the Account class

This program is a modified version of the example (Fig. 3.1)

shown in page 75 in the 10th edition text. \*/

#include <iostream>

#include <string>

#include "Account.h" // need to include the file with the class definition

using namespace std;

int main()

{

Account myAccount; // create Account object myAccount

// show that the initial value of myAccount's name is the empty string

cout << "Initial account name is: " << myAccount.getName();

// prompt for and read name

cout << "\nPlease enter the account name: ";

string theName;

getline(cin, theName); // read a line of text

myAccount.setName(theName); // put theName in myAccount

// display the name stored in object myAccount

cout << "Name in object myAccount is: "

<< myAccount.getName() << endl;

}

1. Class Example with more specific functions (declared inline)

Header file:

/\* Account.h

This class has two private members, has a constructor

with two parameters, as well as two additional functions.

This is a modified form of the class definition

shown in Fig. 3.8 (page 85) in the 10th edition text. \*/

#include <string>

using namespace std;

class Account

{

private:

string name;

int balance{0}; // data member with default initial value

public:

// constructor

Account(string n, int initBal) : name{n} // could have written name = n

{ //here instead of appending it to the function header

// validate that the initBal is greater than 0; if not,

// data member balance keeps its default initial value of 0

if (initBal > 0)

{ // if the initBal is valid

balance = initBal; // assign it to data member balance

}

}

// function that deposits (adds) only a valid amount to the balance

void deposit(int depositAmount)

{

if (depositAmount > 0) // if the depositAmount is valid

{

balance = balance + depositAmount; // add it to the balance

}

}

// function returns the account balance

int getBalance() const

{ return balance; }

// mutator

void setName(string accountName)

{ name = accountName; }

// accessor

string getName() const

{ return name; }

};

Driver program:

/\*

Driver program for the Account class that uses

constructor with two parameters

\*/

#include <iostream>

#include "Account.h"

using namespace std;

int main()

{

Account account1{"Jane Green", 50};

Account account2{"John Blue", -7};

// display initial balance of each object

cout << "account1: " << account1.getName() << " balance is $"

<< account1.getBalance();

cout << "\naccount2: " << account2.getName() << " balance is $"

<< account2.getBalance();

cout << "\n\nEnter deposit amount for account1: "; // prompt

int depositAmount;

cin >> depositAmount; // obtain user input

cout << "adding " << depositAmount << " to account1 balance";

account1.deposit(depositAmount); // add to account1's balance

// display balances

cout << "\n\naccount1: " << account1.getName() << " balance is $"

<< account1.getBalance();

cout << "\naccount2: " << account2.getName() << " balance is $"

<< account2.getBalance();

cout << "\n\nEnter deposit amount for account2: "; // prompt

cin >> depositAmount; // obtain user input

cout << "adding " << depositAmount << " to account2 balance";

account2.deposit(depositAmount); // add to account2 balance

// display balances

cout << "\n\naccount1: " << account1.getName() << " balance is $"

<< account1.getBalance();

cout << "\naccount2: " << account2.getName() << " balance is $"

<< account2.getBalance() << endl;

}