

Section 5.1 Answers to Odd Numbered Questions

1) 67.0 23.1 -17.4 55.2 -5.9

3) 0 4 16

5) 3 1

7)

- a) double temps[100];
- b) double weights[53];
- c) string address[27];

9)

```
for (int x=0; x<MAX_FISHING; x++)
    cout << catches[x] << endl;
```

11)

```
int min=catches[0];
for (int x=1; x<MAX_FISHING; x++)
    if (catches[x] < min)
        min = catches[x];
```

Section 5.1 (Arrays of Structs) Answers to Odd Numbered Questions

1) Dino 12.5
Barney .15

3) Soil dirt[400];

5)

```
for (int a=0; a<400; a++)
    cout << dirt[x].sampleWeight << ' '
        << dirt[x].clayWeight << ' '
        << dirt[x].sandWeight << ' '
        << dirt[x].waterWeight << endl;
```

Section 5.1 (Arrays of Classes) Answers to Odd Numbered Questions

1) 0 2 4

3) Pixel littlePointsOfLight[499];

5)

```
int x, y;
for (int a=0; a<499; a++)
{
    littlePointsOfLight[a].ReturnPair1(x, y);
    cout << x << ',' << y << ' ';
    littlePointsOfLight[a].ReturnPair2(x, y);
    cout << x << ',' << y << ' ';
    littlePointsOfLight[a].ReturnPair3(x, y);
    cout << x << ',' << y << endl;
}
```

Section 5.2 Answers to Odd Numbered Questions

1) 0 0 0 0 0 0 0 0 0 0

3) 0 0 0 0 0 0 0 0 0 0

5) 0 1 2 3 4 5 6 7 8 9

7) 0 1 2
1 2 3

Section 5.3 Answers to Odd Numbered Questions

1) nums[0] -> 17
nums[1] -> 19
nums[2] -> 22
nums[3] -> 23
nums[4] -> -1
nums[5] -> 34

```
3) int aIndex=0, bIndex=0, cIndex=0;
   while (aIndex < aCount && bIndex < bCount && cIndex < MAX_ITEMS)
   {
       if (a[aIndex] < b[bIndex])
       {
           c[cIndex] = a[aIndex];
           aIndex++;
       }
       else
       {
           c[cIndex] = b[bIndex];
           bIndex++;
       }
       cIndex++;
   }

   if (cIndex < MAX_ITEMS)
   {
       if (aIndex < aCount)
       {
           for (int x=aIndex; x<aCount && cIndex<MAX_ITEMS; x++)
           {
               c[cIndex] = a[x];
               cIndex++;
           }
       }
   }
```

```

    if (bIndex < bCount)
    {
        for (int x=bIndex; x<bCount && cIndex<MAX_ITEMS; x++)
        {
            c[cIndex] = b[x];
            cIndex++;
        }
    }
}

```

```

5) for (int x=0; x<numStu; x+=3)
    {
        cout << "Student: " << x
            << " Average: "
            << (basketCases[x]+basketCases[x+1]+basketCases[x+2])/3
            << endl;
    }

```

7) Example 1:

```

LaCucaracha roaches[100];
bool allDead = false;
while (!allDead) {
    for (int x=0; x<100; x++)
        if (roaches[x].IsAlive())
            roaches[x].StompBug();
    for (int x=0; x<100; x++)
        if (roaches[x].IsAlive()){
            roaches[x].ChangeDirection();
            roaches[x].RunLikeHeck();
            cout << x << ' ';
        }
    cout << endl;
    allDead = true;
    for (int x=0; x<100; x++)
        if (roaches[x].IsAlive())
            allDead = false;
}

```

Example 2:

```
LaCucaracha roaches[100];
bool allDead = false;
while (!allDead)
{
    for (int x=0; x<100; x++)
    {
        allDead = true;
        if (roaches[x].IsAlive())
        {
            roaches[x].StompBug();
            if (roaches[x].IsAlive())
            {
                roaches[x].ChangeDirection();
                roaches[x].RunLikeHeck();
                allDead = false;
                cout << x << ' ';
            }
        }
        cout << endl;
    }
}
```

Answers to Programming Projects using Arrays**Online Sales: Example 1**

```

#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>

struct item {                // struct for inventory array
    string description;      // item name
    float  price;           // item price
    int    quantity;        // instock quantity
};

struct selection {          // struct for shopping cart array
    int menuNumber;         // index of selected item in inventory array
    int quantity;          // requested quantity
};

const int MAXSIZE = 20;    // maximum size of arrays

void LoadInventoryArray(item inventory[], int & count);
// pre:  empty inventory array of type item
// post: array inventory contains count number of elements in a list,
//       each element represents an item in inventory

void ShowMenu(item inventory[], int max);
// pre:  array inventory contains max items
// post: the menu is generated from the items in the inventory array

void GenerateReceipt_and_UpdateInventory(item inventory[], selection cart[], int
cartCount);
// pre:  array inventory contains the items in the inventory,
// array cart contains cartCount items that the user has selected to purchase
// post: the customer receipt is generated

void SaveInventoryChanges(item inventory[], int count);
// pre:  array inventory contains the updated inventory of count items
// post: the file "inventory.dat" is re-written with the information in
//       array inventory

int main() {
    item    inventory[MAXSIZE];    // inventory array
    int     invenItems;           // inventory list size
    selection cart[MAXSIZE];       // shopping cart array
    int     cartItems = 0;         // shopping cart size
    int     choice;               // users selection
    int     quantity;             // users requested quantity

    cout.setf(ios::fixed, ios::floatfield);
    cout.setf(ios::showpoint);

    // load the inventory into the inventory array
    LoadInventoryArray(inventory, invenItems);

```

```

// loop until the user chooses to check out or quit
do {
    // output the menu
    ShowMenu(inventory, invenItems);

    // get the selection from the user
    cout << "Enter the item number or "
         << invenItems << " to check out or "
         << invenItems+1 << " to quit." << endl;
    cin >> choice;

    // if the user selects an item to purchase, get the quantity, update
    // the inventory array, then and add the item and quantity to the
    // shopping cart
    if (choice < invenItems && choice >= 0) {
        do {
            cout << "How many of " << inventory[choice].description
                 << " would you like? ";
            cin >> quantity;
        } while (!(quantity >=0 && quantity <= inventory[choice].quantity));
        if (quantity > 0) {
            cart[cartItems].menuNumber = choice;
            cart[cartItems].quantity = quantity;
            inventory[choice].quantity -= quantity;
        }
        cartItems++;
    }
    else if (choice > invenItems+1 || choice < 0)
        cout << "\n\nInvalid Entry\n\n";
} while (choice != invenItems && choice != invenItems+1);

// if the choice was to check out, generate the sales receipt and
// save the inventory data back into the file
if (choice == invenItems) {
    GenerateReceipt_and_UpdateInventory(inventory, cart, cartItems);
    SaveInventoryChanges(inventory, invenItems);
}

return 0;
}

void LoadInventoryArray(item inventory[], int & count)
// pre: empty inventory array of type item
// post: array inventory contains count number of elements in a list,
//       each element represents an item in inventory
{
    ifstream inFile;
    inFile.open("inventory.dat");
    string description;

    count = 0;
    if (!inFile)
        return;
    getline(inFile, description);
    while (inFile) {
        inventory[count].description = description;
        inFile >> inventory[count].price;
        inFile >> inventory[count].quantity;
        getline(inFile,description);
        count++;
    }
}

```

```

        getline(inFile,description);
    }
    inFile.close();
}

void ShowMenu(item inventory[], int max)
// pre:  array inventory contains max items
// post: the menu is generated from the items in the inventory array
{
    cout << endl << "Buy from us!" << endl;
    for (int x=0; x<max; x++) {
        cout << setw(2) << x << " - "
            << inventory[x].description
            << setw(45-inventory[x].description.length()) << " $"
            << setw(10) << setprecision(2) << inventory[x].price
            << " quan. of " << setw(5) << inventory[x].quantity << endl;
    }
    cout << max << "-check our, "
        << max+1 << "-quit" << endl;
}

void GenerateReceipt_and_UpdateInventory(item inventory[],
                                        selection cart[],
                                        int cartCount)
// pre:  array inventory contains the items in the inventory,
// array cart contains cartCount items that the user has selected to purchase
// post: the customer receipt is generated
{
    double total=0;

    cout << "Your Purchased: " << endl;
    for (int x=0; x<cartCount; x++) {
        cout << setw(2) << cart[x].quantity << ' '
            << inventory[cart[x].menuNumber].description
            << " at $" << setprecision(2)
            << inventory[cart[x].menuNumber].price
            << " each for a total of $"
            << setprecision(2)
            << inventory[cart[x].menuNumber].price * cart[x].quantity
            << endl;
        total += inventory[cart[x].menuNumber].price * cart[x].quantity;
    }
    cout << "The total cost of your purchase is $"
        << setprecision(2) << total << endl;
}

void SaveInventoryChanges(item inventory[], int count)
// pre:  array inventory contains the updated inventory of count items
// post: the file "inventory.dat" is re-written with the information in
//       array inventory
{
    ofstream outFile;
    outFile.open("inventory.dat");
    for (int x=0; x<count; x++) {
        outFile << inventory[x].description << endl
            << inventory[x].price << endl
            << inventory[x].quantity << endl;
    }
    outFile.close();
}

```

Online Sales: Example 2

```
#include <iostream>
#include <string>
#include <iomanip>
#include <fstream>
using namespace std;

struct Item
{
    string description;
    float price;
    int qty;
};

struct Selection
{
    int menuNumber;
    int qty;
};

int LoadItems(Item[]);
void DisplayMenu(Item[],int);
void UpdateInv(Item[],int,int);
void UpdateSel(Selection[],int,int,int);
void PrintReceipt(Selection[],Item[],int);
void WriteOutData(Item[],int);
void ClearSel(Selection[]);

const int MAX = 20;

int main()
{
    Item inv[MAX];
    Selection sel[MAX];
    int count=0, cancel=0, changes=0;
    int choice, howMany, num;
    string junk;

    num = LoadItems(inv);
    if (num == -1)
        return 1;

    cout<<setprecision(2);
    cout.setf(ios::fixed,ios::floatfield);
    cout.setf(ios::showpoint);

    do
    {
        DisplayMenu(inv,num);
        cout<<"Choice: ";
        cin>>choice;
        if (choice < num)
        {
            cout<<"Quantity: ";
            cin>>howMany;
            if (inv[choice].qty < howMany)
```

```

        {
            cout<<"SORRY:  Only "<<inv[choice].qty<<" are available."
                <<"  Please try again.\n";
            cout<<"Press <enter> to continue ...";
            getline(cin,junk);
            getline(cin,junk);
            continue;
        }
        UpdateInv(inv,choice,howMany);
        UpdateSel(sel,choice,howMany,count);
        changes = 1;
        count++;
    }
    if (choice == num + 1)
        cancel = 1;

} while (choice < num);

if (changes)
    if (!cancel)
    {
        PrintReceipt(sel,inv,count);
        WriteOutData(inv,num);
    }
    else
    {
        ClearSel(sel);
        cout<<"Transaction Cancelled ... Goodbye!\n\n";
    }
else
    cout<<"No selections ... Oh well, Goodbye!\n\n";

return 0;
}

int LoadItems(Item inv[])
{
    ifstream inFile;
    int    count;
    string  extra;

    inFile.open("inventory.dat");
    if (!inFile)
    {
        cout<<"Input file does not exist!\n";
        return -1;
    }
    count = 0;
    getline(inFile,inv[count].description);
    inFile>>inv[count].price>>inv[count].qty;
    getline(inFile,extra);
    while(inFile && count < MAX)
    {
        count++;
        if (count < MAX)
        {
            getline(inFile,inv[count].description);
            inFile>>inv[count].price>>inv[count].qty;
            getline(inFile,extra);
        }
    }
}

```



```
    for(i=0;i<num;i++)
        outFile<<inv[i].description<<endl
            <<inv[i].price<<endl<<inv[i].qty<<endl;

    outFile.close();
}

void ClearSel(Selection sel[])
{
    int i;

    for(i=0;i<MAX;i++)
    {
        sel[i].menuNumber = -1;
        sel[i].qty = -1;
    }
}
```

Candy Machines**File "machineb.h"**

```

#include <string>

struct Item{
    int count;
    int price;
    string name;
};

const int MAX = 20;

class CandyMachine
{
    public:
        CandyMachine();
        CandyMachine(string initName);
        ~CandyMachine();
        void Menu();
        int GetChoice();
        int Process(int);
        void BuyCandy(int choice);
        int GetMoney(double price);
        int CheckStock(int);
        int ProcessMgr();
        int VerifyPW();
        void CheckMoney();
        void ChangePW();
        void Restock();
    private:
        string name;
        string pw;
        float money;
        Item items[MAX];
        int numItems;
};

```

File "machineb.cc"

```

#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include "machineb.h"

CandyMachine::CandyMachine(){;}

CandyMachine::CandyMachine(string initName)
{
    ifstream inFile;
    name = initName + ".dat";

    inFile.open(name.c_str());
    if (!inFile)
    {
        money = 0.0;
    }
}

```

```

    pw = "password";
    numItems = 0;
}
else
{
    inFile >> pw >> money;
    numItems = 0;
    do
    {
        inFile >> items[numItems].count >> items[numItems].price;
        getline(inFile, items[numItems].name);
        if (inFile)
            numItems++;
        else
            break;
    } while (numItems < MAX);
}
inFile.close();
}

CandyMachine::~CandyMachine()
{
    ofstream outFile;

    outFile.open(name.c_str());
    outFile << pw << endl
        << money << endl;
    for (int x=0; x<numItems; x++)
        outFile << items[x].count << ' '
            << items[x].price << ' '
            << items[x].name << endl;
    outFile.close();
}

void CandyMachine::Menu()
{
    cout<<"\n\nWelcome to Nona's Candy Machine\n\n";
    cout<<"Select one of the following:\n";
    for (int x=0; x<numItems; x++)
        cout << setw(5) << x << " $"
            << setw(5) << setprecision(2) << items[x].price << ' '
            << items[x].name << endl;
}

int CandyMachine::GetChoice()
{
    int choice;
    cout<<"Choice: ";
    cin>>choice;
    return choice;
}

int CandyMachine::Process(int choice)
{
    int status = 0;
    string xtra;

    if (choice == 21)
        status = ProcessMgr();
}

```

```

else
    if (choice < numItems && choice >= 0)
    {
        BuyCandy(choice);
    }
else
{
    cout <<"Invalid Choice -- Try Again\n"
        <<"Press <Enter> to Continue . . .\n";
    cin.get();
}
return status;
}

int CandyMachine::VerifyPW()
{
    string mgrPW, xtra;

    cout<<"This option requires a password.\n";
    cout<<"Please enter password:  ";
    cin>>mgrPW;
    if (mgrPW == pw)
        return 1;

    cout<<"Invalid Password -- Access Denied!\n";
    cout<<"Press C and then <Enter> to Continue . . .\n";
    cin>>xtra;
    return -1;
}

int CandyMachine::ProcessMgr()
{
    int choice;
    string xtra;

    if (VerifyPW() == -1) // Invalid Password
        return 0;

    do
    {
        cout<<"\n\nWelcome to the Manager's Menu\n\n";
        cout<<"Select one of the following:\n";
        cout<<" 1 - Check Total Money Collected\n";
        cout<<" 2 - Change Password\n";
        cout<<" 3 - Restock Machine\n";
        cout<<" 4 - Change Machine\n";
        cout<<" 5 - Return to Main Menu\n\n";
        cout<<"Choice:  ";
        cin>>choice;
        switch(choice)
        {
            case 1: CheckMoney();
                    break;
            case 2: ChangePW();
                    break;
            case 3: Restock();
                    break;
            case 4: cout<<"Choose Different Machine Selected !!!!\n"
                    <<"Good Bye . . . . .\n\n";
                    return 99;
        }
    }
}

```

```

        case 5: return 0;
        default: cout<<"Invalid Choice -- Try Again\n"
                 <<"Press C and then <Enter> to Continue . . .\n";
                 cin>>xtra;
    }

} while (choice != 5);
}

void CandyMachine::BuyCandy(int choice)
{
    if (GetMoney(items[choice].price) == 1 && CheckStock(choice) == 1)
    {
        items[choice].count--;
        money = money + items[choice].price;
    }
}

int CandyMachine::GetMoney(double price)
{
    float moneyIn;
    string xtra;

    cout<<"Please enter your money: ";
    cin>>moneyIn;
    if (moneyIn < price)
    {
        cout<<"Insufficient money -- sale cancelled!\n";
        cout<<"Press C and then <Enter> to Continue . . .\n";
        cin>>xtra;
        return -1;
    }
    else
    {
        if (moneyIn > price)
            cout<<"Your change is: $"<<moneyIn - price<<endl;
        cout<<"Press C and then <Enter> to Continue . . .\n";
        cin>>xtra;
        return 1;
    }
}

int CandyMachine::CheckStock(int choice)
{
    if (items[choice].count < 1)
    {
        cout<<"Item Out of Stock -- sale cancelled!\n";
        cout<<"Press <Enter> to Continue . . .\n";
        cin.get();
        return -1;
    }
    return 1; // Selection is available
}

void CandyMachine::CheckMoney()
{
    cout<<"There is $"<<money<<" in the machine.\n";
    cout<<"Press <Enter> to Continue . . .\n";
    cin.get();
}

```

```

void CandyMachine::ChangePW()
{
    string xtra;

    cout<<"Enter the new password: ";
    cin>>pw;
    cout<<"Password has been changed!\n";
    cout<<"Press <Enter> to Continue . . .\n";
    cin.get();
}

void CandyMachine::Restock()
{
    string xtra;

    for (int x=0; x<numItems; x++)
        items[x].count = 20;
    money = 0.0;
    cout<<"Machine has been restocked!\n";
    cout<<"Press <Enter> to Continue . . .\n";
    cin.get();
}

```

File "candyarray.cc"

```

#include <string>
#include "machineb.h"
using namespace std;

int main()
{
    CandyMachine myCM[4];
    int machine;
    int choice;
    int status = 0;

    myCM[0]=CandyMachine("Chico");
    myCM[1]=CandyMachine("Harpo");
    myCM[2]=CandyMachine("Groucho");
    myCM[3]=CandyMachine("Zeppo");

    do {
        do {
            cout << "Enter Vending Machine: (0,1,2,3,-1 quits): ";
            cin >> machine;
        } while (machine < -1 || machine > 3);

        if (machine == -1) break;

        do {
            myCM[machine].Menu();
            choice = myCM[machine].GetChoice();
            status = myCM[machine].Process(choice);
            if (status == 99)
                break;           // to choose another machine
        } while (true);
    } while (true);

    return 0;
}

```