

## Structures

### ● Problem:

A television quiz show has information about a number of people who want to become contestants. The name of each potential contestant and some personal characteristics are on file. The quiz show producer wants to get an answer to a question such as: "Which contestants have blond hair?" or "Which contestants are 21 years old?"

Write a program to do the following: read in information for a group of contestants and count the number of contestants read in. The information about each contestant consists of:

- name (last, first)
- sex (F or M)
- hair color (red, black, brown, blond, gray, or bald)
- age
- job title
- annual salary (to two decimal places)

e.g., Smith Mary F brown 27 lawyer 87654.32

After reading in all the information, print a table containing all the information for all contestants. The table should have appropriate column headings. A typical row of the table would be:

Mary	Smith	F	brown	27	lawyer	\$87654.32
------	-------	---	-------	----	--------	------------

Now print a menu on the screen which allows the user to select a particular trait which is desired in a contestant. The menu contains the names of all possible traits: age, hair color, salary, sex, and job title. In addition, the menu offers the option of quitting the program.

After identifying the trait desired, prompt the user to enter a value that corresponds to that trait (e.g., 17 for age, M for sex, or 50000 for salary). The program prints a list of all contestants who have the selected trait (for salary, we want all contestants whose salary is greater than or equal to the value requested). The program prints their names (first, last). There should also be a heading indicating what question is being answered. For example:

Contestants whose age is 27

Mary Smith  
Paul Cooper

At this point, the program presents the menu again to allow the user to make another selection. the program continues to process requests until "quit" is chosen from the menu.

● **Pseudocode for the Main Program:**

```
/* first part */  
call readdata() to read in and create the contestant database  
call prettyprint() to print the contestant database  
/* second part */  
while user wishes to continue  
    call printmenu() to print a menu of choices  
    call selecttrait() to respond to a particular choice
```

## Structures

- **Concept of a Structure:**

- All components of an array are of the same type.

- The components (fields) of a **structure** can be of different types. The objects declared within a structure are called its **members**.

- **Declaration of a Structure Prototype:**

```
struct struct_name {  
    data_type_1 ident_1;           //member variables  
    data_type_2 ident_2;           //or data members  
    ...  
    data_type_n ident_n;  
};
```

- **Declaration of Variables whose Data Type is a Structure:**

```
struct_name object_1, object_2, ...;
```

- **Example:**

```
struct address {  
    int housenumber;  
    string streetname;  
};
```

```
address home_address, work_address;
```

- **Accessing the Elements of a Structure:**

```
structurename.membername
```

```
home_address.housenumber = 123;  
home_address.streetname = "Main St.";  
cin >> work_address.housenumber;  
getline(cin, work_address.streetname);
```

- **Structure Assignment:**

If two structures are of the same struct type one can assign the value of one structure to the other.

```
work_address = home_address;
```

- **Example:**

```
struct student {  
    string name;  
    double average;  
    char lettergrade;  
};
```

```
student freshman, beststudent;
```

```
...
```

```
if (freshman.average > 3.5)  
    freshman.lettergrade = 'A';
```

```
if (freshman.average > beststudent.average)  
{  
    beststudent.name = freshman.name;  
    beststudent.average = freshman.average;  
    beststudent.lettergrade = freshman.lettergrade;  
}
```

```
or
```

```
if (freshman.average > beststudent.average)  
    beststudent = freshman;
```

- **Initializing a Structure:**

```
struct student {  
    string name;  
    double average;  
    char lettergrade;  
};
```

```
student freshman = {"Sam Starter", 2.0, 'C'};  
student sophomore = {"Joe Later", 3.45};
```

- **A Structure with Member Element Arrays:**

```
struct triangle {  
    string type;  
    int angle[3];  
};
```

```
triangle t1;
```

```
...
```

```
if (t1.angle[0] == t1.angle[1] &&  
    t1.angle[1] == t1.angle[2])  
    t1.type = "equilateral";  
else if (t1.angle[0] == t1.angle[1] ||  
         t1.angle[1] == t1.angle[2] ||  
         t1.angle[0] == t1.angle[2])  
    t1.type = "isosceles";  
else  
    t1.type = "scalene";
```

- **A Structure with Many Elements:**

```
struct books {  
    string lastname;  
    string firstname;  
    string title;  
    string pubname;  
    string pubcity;  
    string pubstate;  
    int yearpub;  
    string call_number;  
    int numcopies;  
};
```

```
books book;
```

## Nested Structures

- **Declaration of a Nested Structure:**

```
struct name {  
    string last;  
    string first;  
};
```

```
struct pub_info {  
    string name;  
    string city;  
    string state;  
};
```

```
struct books {  
    name author;  
    string title;  
    pub_info publisher;  
    int yearpub;  
    string call_number;  
    int numcopies;  
};
```

```
books book;
```

- **Accessing Members of a Nested Structure:**

```
book.publisher.name = "Prentice-Hall";  
book.author.last = "Harrow";  
book.yearpub = 2007;           // not nested  
cout << book.author.first << " " << book.author.last <<  
    endl;
```

## An Array of Structures

- **Example:**

```
struct name {
    string last;
    string first;
};
```

```
struct str_addr {
    int housenum;
    string street;
};
```

```
struct addr {
    str_addr street_address;
    string city;
    string state;
    string zip;
};
```

```
struct employee {
    name empname;
    string socsecnum;
    addr address;
};
```

```
employee emp[100];  
....
```

```
cout << emp[0].address.zip << endl << endl;  
for (int i = 0; i < 10; i++)  
    cout << emp[i].address.street_address.housenum <<  
        " " << emp[i].address.street_address.street << endl;  
cout << endl << "All employees who live in New Jersey:"  
    << endl;  
for (int i = 0; i < 100; i++)  
    if (emp[i].address.state == "NJ")  
        cout << emp[i].empname.last << ", " <<  
            emp[i].empname.first << endl;
```

● **Example of Arrays at Multiple Levels:**

```
struct name {  
    string last;  
    string first;  
};
```

```
struct classmark {  
    int test[5];  
    double average;  
    char lettergrade;  
};
```

```
struct student3 {  
    name sname;  
    int numclasses;  
    classmark class[5];  
    double overallavg;  
};
```

```
student3 stu;
```

```
int sum;  
double sum_classavg = 0.0;  
...
```

```
for (int i = 0; i < stu.numclasses; i++)  
{  
    sum = 0;  
    for (int j = 0; j < 5; j++)  
        sum += stu.class[i].test[j];  
    stu.class[i].average = (double)sum / 5;  
    sum_classavg += stu.class[i].average;  
}  
stu.overallavg = sum_classavg / stu.numclasses;  
cout << stu.sname.last << ", " << stu.sname.first <<  
    " " << stu.overallavg << endl;
```

## Using a Structure in a Function

- **Example:**

```
/* program to print components of employee structure */
#include <iostream>
#include <string>
using namespace std;

struct name {
    string last;
    string first;
};
struct str_addr {
    int housenum;
    string street;
};
struct addr {
    str_addr street_address;
    string city;
    string state;
    string zip;
};
struct employee {
    name empname;
    string socsecnum;
    addr address;
};
void print_emp(employee);           // Function Prototype

int main ()
{
    employee worker;
    ...                               // read in worker info
    print_emp(worker);
    return 0;
}
void print_emp(employee worker)    // passed by value
{
    cout << worker.empname.last << ", " << worker.empname.first
         << " " << worker.socsecnum << endl;
    cout << worker.address.street_address.housenum <<
         " " << worker.address.street_address.street << endl;
    cout << worker.address.city << " " << worker.address.state
         << " " << worker.address.zip << endl;
    return;
}
```

## Changing a Structure in a Function

- **Example:**

```
/* program to read in & print employee structure */
#include <iostream>
#include <string>
using namespace std;
...
struct employee {
    ...
};

/* Function Prototypes */
void read_emp(employee &);
void print_emp(employee);

int main ()
{
    employee worker;

    read_emp(worker);
    print_emp(worker);
    return 0;
}

void print_emp(employee worker)           // pass by value
{
    ...
}

void read_emp(employee &worker)         // pass by reference
{
    cin >> worker.empname.last;
    cin >> worker.empname.first;
    cin >> worker.socsecnum;
    cin >> worker.address.street_address.housenum;
    cin >> worker.address.street_address.street;
    ...
    return;
}
```

## Sending an Array of Structures as a Parameter

- **Example:**

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

struct votes {
    string name;
    int numvotes;
};

/* Function Prototype */
void readvotes(votes [], int &);

int main()
{
    votes candidates[100];
    int numcands;

    readvotes(candidates,numcands);

    return 0;
}

void readvotes(votes cand[], int &numcands)
{
    cin >> numcands;
    for (int i = 0; i < numcands; i++)
    {
        cin >> cand[i].name;
        cin >> cand[i].numvotes;
        cout << cand[i].name << endl;
        cout << cand[i].numvotes << endl;
    }
    return;
}
```

## A Function which Returns a Structure

- **Example:**

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

struct votes {
    string name;
    int numvotes;
};

/* Function Prototype */
votes whowon(votes [], int);           // returns a struct
void readvotes(votes [], int &);

int main()
{
    votes cand[100], winner;
    int numcands;

    readvotes(cand, numcands);
    winner = whowon(cand, numcands);
    cout << winner.name << " won with " << winner.numvotes
         << " votes" << endl;
    return 0;
}
...
votes whowon(votes v[], int numcands)
{
    votes highest;

    highest = v[0];
    for (int i = 1; i < numcands; i++)
    {
        if (v[i].numvotes > highest.numvotes)
            highest = v[i];
    }
    return (highest);
}
```

## Return to the Problem

- **Pseudocode for the Main Program:**

```
/* first part */  
call readdata() to read in and create the contestant database  
call prettyprint() to print the contestant database  
/* second part */  
while user wishes to continue  
    call printmenu() to print a menu of choices  
    call selecttrait() to respond to a particular choice
```

- **Declaration for the Contestant Database:**

```
const int NUMCONS = 50;
```

```
struct conname {  
    string last;  
    string first;  
};  
struct jobinfo {  
    string title;  
    double salary;  
};  
struct persinfo {  
    char sex;  
    string haircolor;  
    int age;  
    jobinfo job;  
};  
struct con {  
    conname name;  
    persinfo personal;  
};
```

```
...
```

```
con contestant[NUMCONS];
```

- **The Main Program:**

```
/* Contestant Database */
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

const int NUMCONS = 50;

/* structure definitions go here */
...

/* Function Prototypes */
void readdata(con [], int &);
void prettyprint(con [], int);
void printmenu();
int selecttrait(con [], int);

int main()
{
    con contestant[NUMCONS];
    int num;

    /* first part */
    /* fill and print database */
    readdata(contestant,num);
    prettyprint(contestant,num);

    /* second part */
    /* call functions to read and process requests */
    do {
        printmenu();
    } while (selecttrait(contestant,num) != 0);

    return 0;
}
```

● **The Function readdata():**

```
/* Function readdata() */
void readdata(con contestant[], int &count)
{
    // open input file
    ifstream cfile("c:\\mypgms\\myinput.txt");
//    ifstream cfile("con");          //un-comment for debugging

    count = 0;                          //initialize count

    while (cfile >> contestant[count].name.last)
    {
        cfile >> contestant[count].name.first;
        cfile >> contestant[count].personal.sex;
        cfile >> contestant[count].personal.haircolor;
        cfile >> contestant[count].personal.age;
        cfile >> contestant[count].personal.job.title;
        cfile >> contestant[count].personal.job.salary;
        count++;
    }

    cfile.close();
    return;
}
```

- **The Function prettyprint():**

```
/* Function prettyprint: */
void prettyprint(con contestant[], int num)
{
    // open output file
    ofstream dbfile("c:\\mypgms\\myoutput.txt");
//    ofstream dbfile("con");    //un-comment for debugging

    dbfile << "\t\tContestants in the Database\n\n";
    dbfile << "Name\t\tSex\tHair\tAge\tTitle\tsalary\n\n";
    for (int count = 0; count < num; count + +)
    {
        dbfile << contestant[count].name.first;
        dbfile << " " << contestant[count].name.last;
        dbfile << "\t" << contestant[count].personal.sex;
        dbfile << "\t" << contestant[count].personal.haircolor;
        dbfile << "\t" << contestant[count].personal.age;
        dbfile << "\t" << contestant[count].personal.job.title;
        dbfile << "\t" << contestant[count].personal.job.salary;
        dbfile << endl;
    }

    dbfile.close();
    return;
}
```

- **The Function printmenu():**

```
/* Function printmenu() */
void printmenu()
{
    cout << "\n\n\n\n\n\n\n\n";
    cout << "To obtain a list of contestants with a given\n";
    cout << "trait, select a trait from the list and type in\n";
    cout << "the number corresponding to that trait.\n\n";
    cout << "To quit, select 0.\n\n";
    cout << "\t*****\n";
    cout << "\t  List of Choices\n";
    cout << "\t*****\n";
    cout << "\t  0 -- quit\n";
    cout << "\t  1 -- age\n";
    cout << "\t  2 -- sex\n";
    cout << "\t  3 -- hair color\n";
    cout << "\t  4 -- title\n";
    cout << "\t  5 -- salary\n";
    cout << "\n\n\tEnter your selection, 0-5: ";
    return;
}
```

- The Function `selecttrait()`:

```
/* Function selecttrait() */
int selecttrait(con contestant[], int num)
{
    int choice;

    do {
        cin >> choice;
        switch(choice)
        {
            case 0:
                break;
            case 1:
                findage(contestant,num);
                break;
            case 2:
                findsex(contestant,num);
                break;
            case 3:
                findhair(contestant,num);
                break;
            case 4:
                findtitle(contestant,num);
                break;
            case 5:
                findsalary(contestant,num);
                break;
            default:
                cout << "Incorrect value; try again\n";
                cout << "\n\tEnter your selection, 0-5: ";
                break;
        }
    } while (choice < 0 || choice > 5);
    return (choice);
}
```

- **The Function findage():**

```
/* Function findage() */
void findage(con contestant[], int num)
{
    int agewanted,found = 0;

    cout << "\n\nEnter the age you want: ";
    cin >> agewanted;
    cout << "\nContestants whose age is " << agewanted << "\n\n";
    for (int count = 0; count < num; count + +)
        if (contestant[count].personal.age == agewanted)
        {
            cout << contestant[count].name.first << " " <<
                contestant[count].name.last << endl;
            found + +;
        }
    if (!found)
        cout << "No contestants of this age\n\n";
    else
        cout << endl << found << " contestants found\n";

    // give user a chance to look at output
    // before printing menu
    pause();

    return;
}
```

- **The Function pause():**

```
/* Function pause() */
void pause()
{
    system("pause");
    return;
}
```

● **Revised Main Program:**

```
/* Contestant Database */
/* includes go here */
...
const int NUMCONS = 50;
/* structure definitions go here */
...

/* Function Prototypes */
void readdata(con [], int &);
void prettyprint(con [], int);
void printmenu();
int selecttrait(con [], int);
void findage(con [], int);
void findsex(con [], int);
void findhair(con [], int);
void findtitle(con [], int);
void findsalary(con [], int);
void pause();

void main()
{
    con contestant[NUMCONS];
    int num;

    /* first part */
    /* fill and print database */
    readdata(contestant,num);
    prettyprint(contestant,num);

    /* second part */
    /* call functions to read and process requests */
    do {
        printmenu();
    } while (selecttrait(contestant,num) != 0);

    return 0;
}
```

## Classes Declared with class

- By default, unless we explicitly specify otherwise, all the members of a **struct** are **public**. This means all members can be accessed with the **dot operator**
- The keyword **class** can be used in place of struct to declare a class. By default, when **class** is used, all members are **private**. Aside from certain exceptions, all private members are **hidden** and can not be accessed even with the dot operator.
- The keywords **private** and **public** allow us to declare some class members private and others public. All declarations following one of these keywords will be private or public, respectively, until another such keyword is encountered.

```
class con {  
    //private member variables  
    conname name;  
    persinfo personal;  
};
```

```
class con {  
    public:  
    //public member variables  
    conname name;  
    persinfo personal;  
};
```

```
class con {  
    public:  
    conname name;           //public member variable  
    private:  
    persinfo personal;     //private member variable  
};
```

● **Revised Main Program - Using class - Public Variables:**

```
/* Contestant Database */
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

//constant definitions
const int NUMCONS = 50;

//class definitions
class conname {
public:
    string last;
    string first;
};

class jobinfo {
public:
    string title;
    double salary;
};

class persinfo {
public:
    char sex;
    string haircolor;
    int age;
    jobinfo job;
};

class con {
public:
    conname name;
    persinfo personal;
};

//function prototypes
...

int main()
...
```

## Member Functions

- C++ allows classes to contain functions as members. These **member functions** are used to send messages and receive replies from objects declared using a class. Member functions designed to provide a user interface to an object should be declared **public**.
- **Example:**

```
class con {  
    // private member variables  
    conname name;  
    persinfo personal;  
    public:  
    // public member functions  
    bool readdata(istream &);  
    void prettyprint(ofstream &);  
    bool compareage(int);  
};
```

- The member functions `readdata()`, `prettyprint()`, and `compareage()` can be used to manipulate objects of type `con`.

```
con contestant[NUMCONS];
```

```
count = 0;  
while (contestant[count].readdata(cfile))  
    count++;
```

```
for (int count = 0; count < num; count++)  
    contestant[count].prettyprint(dbfile);
```

```
found = 0;  
for (int count = 0; count < num; count++)  
    if (contestant[count].compareage(agemwanted))  
        found++;
```

● **Member Function readdata():**

```
/* con member function readdata():
 * Input:
 *   cfile - a reference to the input file
 * Process:
 *   read the input file and load the object's data members
 * Output:
 *   return true if file is read and object fields loaded
 *   else return false if EOF reached.
 */
bool con::readdata(ifstream &cfile)
{
    if (cfile >> name.last)
    {
        cfile >> name.first;
        cfile >> personal.sex;
        cfile >> personal.haircolor;
        cfile >> personal.age;
        cfile >> personal.job.title;
        cfile >> personal.job.salary;
        return true;
    }
    return false;
}
```

## ● Member Function prettyprint():

```
/* con member function prettyprint():
 * Input:
 *   dbfile - a reference to the output file
 * Process:
 *   write the object's data members to the output file
 * Output:
 *   the written data members of the object
 */
void con::prettyprint(ofstream &dbfile)
{
    dbfile.setf(ios::fixed,ios::floatfield);
    dbfile.precision(2);                //set decimal precision

    dbfile << name.first;
    dbfile << " " << name.last;
    dbfile << "\t" << personal.sex;
    dbfile << "\t" << personal.haircolor;
    dbfile << "\t" << personal.age;
    dbfile << "\t" << personal.job.title;
    dbfile << "\t";
    dbfile.width(9);
    dbfile << personal.job.salary;
    dbfile << endl;

    return;
}
```

● **Member Function compareage():**

```
/* con member function compareage():
 * Input:
 *   agewanted - age wanted
 * Process:
 *   compare agewanted to the object's age
 * Output:
 *   if ages are the same, print the object's name
 *   and return true
 *   else return false
 */
bool con::compareage(int agewanted)
{
    if (personal.age == agewanted)
    {
        cout << name.first << " " << name.last << endl;
        return true;
    }
    return false;
}
```

● **Revised Main Program - Using class - Private Variables:**

```
/* Contestant Database - with member functions */
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

//constant definitions
const int NUMCONS = 50;

//class definitions
class conname {
public:
    string last;
    string first;
};

class jobinfo {
public:
    string title;
    double salary;
};

class persinfo {
public:
    char sex;
    string haircolor;
    int age;
    jobinfo job;
};

class con {
    // member variables                //private variables
    conname name;
    persinfo personal;
public:
    // member functions                //public functions
    bool readdata(ifstream &cfile);
    void prettyprint(ofstream &dbfile);
    bool compareage(int agewanted);
};
```

```

// Function Prototypes
void readdata(con [], int &);
void prettyprint(con [], int);
void printmenu(void);
int selecttrait(con [], int);
void findage(con [], int);
void findsex(con [], int);
void findhair(con [], int);
void findtitle(con [], int);
void findsalary(con [], int);
void pause(void);

int main()
{
    con contestant[NUMCONS];
    int num;

    /* first part */
    /* fill and print database */
    readdata(contestant,num);
    prettyprint(contestant,num);

    /* second part */
    /* call functions to read and process requests */
    do {
        printmenu();
    } while (selecttrait(contestant,num) != 0);

    return 0;
}

```

- **Revised Function readdata():**

```
void readdata(con contestant[], int &count)
{
    // open input file
    ifstream cfile("c:\\mypgms\\myinput.txt");
//    ifstream cfile("con");           //un-comment for debugging

    count = 0;           //initialize count

    while (contestant[count].readdata(cfile))
        count + +;

    cfile.close();
    return;
}
```

- **Revised Function prettyprint():**

```
void prettyprint(con contestant[], int num)
{
    // open output file
    ofstream dbfile("c:\\mypgms\\myoutput.txt");
//    ofstream dbfile("con");           //un-comment for debugging

    dbfile << "\t\tContestants in the Database\n\n";
    dbfile << "Name\t\tSex\tHair\tAge\tTitle\tsalary\n\n";

    for (int count = 0; count < num; count + +)
        contestant[count].prettyprint(dbfile);

    dbfile.close();
    return;
}
```

- **Revised Function findage():**

```
void findage(con contestant[], int num)
{
    int agewanted,found = 0;

    cout << "\n\nEnter the age you want: ";
    cin >> agewanted;
    cout << "\nContestants whose age is " << agewanted << "\n\n";
    for (int count = 0; count < num; count + +)
        if (contestant[count].compareage(agewanted))
            found + +;
    if (!found)
        cout << "No contestants of this age\n\n";
    else
        cout << endl << found << " contestants found\n\n";

    // give user a chance to look at output
    // before printing menu
    pause();

    return;
}
```