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 blonde 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.

- **Declaration of a Structure Prototye:**

```
struct struct_tag {
    data_type_1 ident_1;
    data_type_2 ident_2;
    ...
    data_type_n ident_n;
};
```

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

```
struct struct_tag struct_var_1,struct_var_2,...;
```

- **Example:**

```
struct address {
    int housenumber;
    char streetname[20];
};

struct address home_address, work_address;
```

- **Alternate Declaration of a Structure and Structure Variables:**

```
struct struct_tag {
    data_type_1 ident_1;
    data_type_2 ident_2;
    ...
    data_type_n ident_n;
} struct_var_1, struct_var_2;
```

- **Example:**

```
struct address {
    int housenumber;
    char streetname[20];
} home_address, work_address;
```

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

structurename.membername

```
home_address.housenumber = 123;
strcpy(home_address.streetname, "Main");
scanf("%d",&work_address.housenumber);
scanf("%s",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 {
    char name[40];
    double average;
    char lettergrade;
} freshman;

struct student beststudent;

...

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

if (freshman.average >  beststudent.average) {
    strcpy(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 {
    char name[40];
    double average;
    char lettergrade;
} freshman =  {"Sam Starter", 2.0, 'C'};

struct student sophomore =  {"Joe Later", 3.45};
```

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

```
struct triangle {
    char  type[12];
    int   angle[3];
} t1;

...

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

- **A Structure with Many Elements:**

```
struct books {
    char  lastname[20];
    char  firstname[20];
    char  title[30];
    char  pubname[25];
    char  pubcity[20];
    char  pubstate[3];
    int   yearpub;
    char  call_number[15];
    int   numcopies;
} book;
```

Nested Structures

- **Declaration of a Nested Structure:**

```
struct name {
    char last[20];
    char first[20];
};

struct pub_info {
    char name[25];
    char city[20];
    char state[3];
};

struct books {
    struct name author;
    char  title[30];
    struct pub_info publisher;
    int    yearpub;
    char  call_number[15];
    int    numcopies;
} book;
```

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

```
strcpy(book.publisher.name,"Prentice-Hall");
strcpy(book.author.last,"Harrow");
book.yearpub = 1996;          // not nested
printf("%s %s", book.author.first, book.author.last);
```

An Array of Structures

- **Example:**

```
struct name {
    char last[20];
    char first[20];
};
struct str_addr {
    int housenum;
    char street[20];
};
struct addr {
    struct str_addr street_address;
    char city[30];
    char state[3];
    char zip[6];
};
struct employee {
    struct name empname;
    char socsecnum[10];
    struct addr address;
} emp[100];

....

printf("%s\n\n",emp[0].address.zip);
for (i = 0; i < 10; i++ )
    printf("%d %s\n",
        emp[i].address.street_address.housenum,
        emp[i].address.street_address.street);
printf("\nAll employees who live in New Jersey:\n");
for (i = 0; i < 100; i++ )
    if (strcmp(emp[i].address.state,"NJ") == 0)
        printf("%s %s\n",emp[i].empname.last,
            emp[i].empname.first);
```

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

```
struct name {
    char last[20];
    char first[20];
};
struct classmark {
    int test[5];
    double average;
    char lettergrade;
};
struct student3 {
    struct name sname;
    int numclasses;
    struct classmark class[5];
    double overallavg;
} stu;

int i,j,k,sum;
double sum_classavg =  0.0;

...

for (i =  0; i <  stu.numclasses; i+ + ) {
    sum =  0;
    for (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;
printf("%s, %s %8.2f\n", stu.sname.last, stu.sname.first,
        stu.overallavg);
```

Pointer to a Structure

- **Example of a Pointer to a Structure:**

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

struct student3 * studptr;      // pointer to a student3 structure

struct student3 stud;           // a student3 structure
```

- **Usage:**

```
studptr = &stud;
(* studptr).numclasses = 5;
(* studptr).overallavg = 3.15;
printf("%d\n", (* studptr).numclasses);
printf("%s\n", (* studptr).sname.last);
scanf("%d", &(* studptr).numclasses);
scanf("%s", (* studptr).sname.last);
```

- **The -> Operator:**
  ***ptr* -> means (* ptr).**

```
studptr -> numclasses = 5;
studptr -> overallavg = 3.15;
printf("%d\n", studptr -> numclasses);
printf("%s\n", studptr -> sname.last);
scanf("%d", &studptr -> numclasses);
or scanf("%d", &(studptr -> numclasses));
scanf("%s", studptr -> sname.last);
```

Using a Structure in a Function

- **Example:**

```
/*  program to print components of employee structure * /
#include < stdio.h>
struct name {
    char last[20];
    char first[20];
};
struct str_addr {
    int housenum;
    char street[20];
};
struct addr {
    struct str_addr street_address;
    char city[30];
    char state[3];
    char zip[6];
};
struct employee {
    struct name empname;
    char socsecnum[10];
    struct addr address;
};
void print_emp(struct employee);        // Function Prototype

void main ()
{
    struct employee worker;
    ...                                 // read in worker info
    print_emp(worker);
}
void print_emp(struct employee worker)      // passed by value
{
    printf("%s %s\n",worker.empname.last, worker.empname.first);
    printf("%s\n",worker.socsecnum);
    printf("%d %s\n", worker.address.street_address.housenum,
        worker.address.street_address.street);
    printf("%s %s\n",worker.address.city, worker.address.state);
    return;
}
```

Changing a Structure in a Function

- **Example:**

```
/* program to read in & print employee structure */
#include < stdio.h>
   ...
struct employee {
   ...
};

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

void main ()
{
   struct employee worker;

   read_emp(&worker);
   print_emp(worker);
}

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

void read_emp(struct employee * worker)    // pass by reference
{
   scanf("%s", worker -> empname.last);
   scanf("%s", worker -> empname.first);
   scanf("%s", worker -> socsecnum);
   scanf("%d", &worker -> address.street_address.housenum);
   scanf("%s", worker -> address.street_address.street);
   ...
   return;
}
```

Sending an Array of Structures as a Parameter

- **Example:**

```
#include < stdio.h>
struct votes {
    char name[20];
    int numvotes;
};

/* Function Prototype */
void readvotes(struct votes *, int * );

void main()
{
    struct votes cands[100];
    int numcands;

    readvotes(cands,&numcands);
}

void readvotes(struct votes * cand, int * numcands)
{
    int i;

    scanf("%d", numcands);
    for (i = 0; i < * numcands; i++ ) {
        scanf("%s", cand[i].name);
        scanf("%d", &cand[i].numvotes);
        printf("%s\n", cand[i].name);
        printf("%d\n", cand[i].numvotes);
    }
    return;
}
```

- **Alternative Method Using Pointer Notation:**

```
    scanf("%s", (cand+ i) -> name);   // must use parentheses
    scanf("%d", &(cand+ i) -> numvotes);
```

A Function which Returns a Structure

- **Example:**

```
#include < stdio.h>
#include < string.h>
struct votes {
   char name[20];
   int numvotes;
};

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

void main()
{
   struct votes cands[100],winner;
   int numcands;

   readvotes(cands,&numcands);
   winner =  whowon(cands,numcands);
   printf("%s won with %d votes\n",winner.name,winner.numvotes);
}
...

struct votes whowon(struct votes * v, int numcands)
{
   int i;
   struct votes highest;

   highest =  v[0];
   for (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:**

```
#define NUMCONS 50
struct conname {
    char last[20];
    char first[20];
};
struct jobinfo {
    char title[15];
    double salary;
};
struct persinfo {
    char sex[2];
    char haircolor[7];
    int age;
    struct jobinfo job;
};
struct con {
    struct conname name;
    struct persinfo personal;
};

...

struct con contestant[NUMCONS];
```

- **The Main Program:**

```
/* Contestant Database */
#include < stdio.h>
#include < stdlib.h>
#include < string.h>
#define NUMCONS 50

/* structure definitions go here */
...

/* Function Prototypes */
void readdata(struct con *, int *);
void prettyprint(struct con *, int);
void printmenu(void);
int selecttrait(struct con *, int);

void main()
{
   struct 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);
}
```

- **The Function readdata():**

```
/* Function readdata() */
void readdata(struct con *contestant, int *num)
{
    FILE *cfile;
    int count = 0;

    cfile = fopen("input.dat","r");
    if (cfile == NULL) {
        fprintf(stderr,"Error opening input file\n");
        exit(1);
    }
    while (fscanf(cfile,"%s",contestant[count].name.last != EOF)) {
        fscanf(cfile,"%s",contestant[count].name.first);
        fscanf(cfile,"%s",contestant[count].personal.sex);
        fscanf(cfile,"%s",contestant[count].personal.haircolor);
        fscanf(cfile,"%d",&contestant[count].personal.age);
        fscanf(cfile,"%s",contestant[count].personal.job.title);
        fscanf(cfile,"%lf",&contestant[count].personal.job.salary);
        count++;
    }
    *num = count;
    fclose(cfile);
    return;
}
```

- **The Function prettyprint():**

```
/* Function prettyprint: */
void prettyprint(struct con *contestant, int num)
{
    FILE *dbfile;
    int count = 0;

    dbfile = fopen("output.dat","w");
    if (dbfile = = NULL) {
        fprintf(stderr,"Error opening output file\n");
        exit(1);
    }
    fprintf(dbfile,"\t\tContestants in the Database\n\n");
    fprintf(dbfile,"Name\t\tSex\tHair\tAge\tTitle\tsalary\n\n");
    for (count = 0; count < num; count+ + ) {
        fprintf(dbfile,"%s",contestant[count].name.first);
        fprintf(dbfile,"%s\t",contestant[count].name.last);
        fprintf(dbfile,"%s\t",contestant[count].personal.sex);
        fprintf(dbfile,"%s\t",contestant[count].personal.haircolor);
        fprintf(dbfile,"%d\t",contestant[count].personal.age);
        fprintf(dbfile,"%s\t",contestant[count].personal.job.title);
        fprintf(dbfile,"%lf\n",contestant[count].personal.job.salary);
    }
    fclose(dbfile);
    return;
}
```

- **The Function printmenu():**

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

- **The Function selecttrait():**

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

    do {
        fscanf(stdin,"%d",&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:
                fprintf(stdout,"Incorrect value; try again\n");
                fprintf(stdout,"\n\tEnter your selection, 0-5: ");
                break;
        }
    } while (choice <  0 || choice >  5);
    return (choice);
}
```

- **The Function findage():**

```
/* Function findage() */
void findage(struct con * contestant, int num)      // add to main
{
    int agewanted,count,found= 0;

    fprintf(stdout,"\n\nEnter the age you want: ");
    fscanf(stdin,"%d",&agewanted);
    fprintf(stdout,"\nContestants whose age is %d\n\n",agewanted);
    for (count =  0; count <  num; count++ )
        if (contestant[count].personal.age = =  agewanted) {
            fprintf(stdout,"%s %s\n", contestant[count].name.first,
                contestant[count].name.last);
            found++ ;
        }
    if (!found)
        fprintf(stdout,"No contestants of this age\n\n");
    else
        fprintf(stdout,"\n%d contestants found\n",found);

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

    return;
}
```

- **The Function pause():**

```
/* Function pause() */
void pause(void)
{
    getchar();                    // clear buffer
    fprintf(stdout,"\n\nPress < Enter>  to continue: ");
    getchar();
    return;
}
```

- **Revised Main Program:**

```
/*  Contestant Database */
/*  includes go here */
...
#define NUMCONS 50
/*  structure definitions go here */
...

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

void main()
{
   struct 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);
}
```

Using typedef

- C allows a user to define their own data types.

- **General Form of a typedef Definition:**
  typedef *definition newtype*;

- **Example:**

```
typedef int bigarray[1000];
typedef char string[80];
typedef int * intptr;
...

bigarray nums;          // equivalent to int nums[1000];
string buffer;          // equivalent to char buffer[80];
intptr p;               // equivalent to int * p;
```

- **General Form of a typedef Definition for a Structure:**

```
typedef struct tag {
    type_1 id_1;
    type_2 id_2;
    ...
    type_n id_n;
} typename;
```

- **Example:**

```
typedef struct addr {
    int    housenum;
    char  streetname[10];
} address;

address home_address,work_address;
struct addr new_address;          // either way is acceptable
```

- **Location of typedef Definitions:**
  A type must be defined before it is used!

Unions

- **Purpose of a Union:**
  A **union** allows a single storage location to be associated with several different variable names.

- **General Form of a Union:**

```
union union_tag {
    type_1 var_1;
    type_2 var_2;
    ...
    type_n var_n;
} union_var;
```

- **Example:**

```
union occupation {
    char college[30];
    double salary;
} current, past;                // declare two variables

union occupation *old;          // declares a pointer to a union

...

printf("%s\n",current.college);
old =  &past;
old ->  salary =  12345.67;
```

- **Using a Structure the Contains a Union:**

```
struct alumni {
    char name[30];
    char address[40];
    char citystate[30];
    int incollege
    union occupation {
        char college[30];
        double salary;
    } current;
} grad[100];

...

if (grad[0].incollege)
    printf("%s\n",grad[0].current.college);
else
    printf("%9.2f\n",grad[0].current.salary);
```

- **Using a Union that Contains a Structure:**

```
union showinfo {
    char date_avail[9];
    char date_sched[9];
    struct past_con {
        char date_appeared[9];
        double amount_won;
        char socsecnum[10];
    } onshow;
} state;
```

- **Using a Union in the Previous Program:**

```
#define PROSPECTIVE 1
#define SCHEDULED 2
#define PAST 3

...

struct con {
    struct conname name;
    struct persinfo personal;
    int status;                 // PROSPECTIVE,SCHEDULED,PAST
    union showinfo state;
};
struct contestant[NUMCONS];

...

if (contestant[0].status = =  PAST)
    printf("%s %f %s\n", contestant.state.onshow.date_appeared,
        contestant.state.onshow.amount_won,
        contestant.state.onshow.socsecnum);
else if (contestant[0].status = =  PROSPECTIVE)
    printf("%s\n", contestant.state.date_available);
else if (contestant[0].status = =  SCHEDULED)
    printf("%s\n", contestant.state.date_scheduled);
```