CSC 4630

Supplementary Material on C

Review of Structs
Structures Overview

- What is a structure?
  - A "record"
  - A collection of one or more variables, possibly of different types, grouped under a single name.
  - A derived data type

- Advantages of structures?
  - Used to organize data
  - Allows a group of related variables to be treated as a unit.

<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
<th>salary</th>
</tr>
</thead>
</table>
struct student {
    char    name [10];
    int     student_id;
    char    grade;
} s1, s2;

s1:

    name
    student_id
    grade

s2:

    name
    student_id
    grade
struct student {
    char    name [10];
    int    student_id;
    char    grade;
} s1, s2;

s1.grade = 'A';
s1.student_id = 590017
strcpy(s1.name, "Busker");

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Bushker</td>
</tr>
<tr>
<td>student_id</td>
<td>590017</td>
</tr>
<tr>
<td>grade</td>
<td>A</td>
</tr>
</tbody>
</table>
#define CLASS_SIZE 50
#define NCOURSES 10

struct student {
    char  *last_name;
    int   student_id;
    char  grade;
};

struct date {
    short day;
    char  month[10];
    short year;
};

struct personal {
    char   name [20];
    struct date birthday;
};

struct student_data {
    struct personal p;
    int          student_id;
    char         grade[NCOURSES];
};
Alternative Ways of Declaring Structures

1. Establish tag and declare variables in same place:
   
   ```
   struct emp {
       char   name[21];
       char   id[8];
       double salary;
   } prgmr, employee[100], *p;
   ```

2. Leave out the tag:
   
   ```
   struct {
       char   name[21];
       char   id[8];
       double salary;
   } prgmr, employee[100], *p;
   ```

3. Use a typedef:
   
   ```
   typedef struct {
       char   name[21];
       char   id[8];
       double salary;
   } EMPLOYEE;

   EMPLOYEE prgmr, employee[100], *p;
   ```
Renaming a Type - typedef

typedef existing_type new_type

- C statement - new name for a data type
- Often found in header files
- Readability and Portability
- Syntax like variable declaration
- Upper-case recommended

1 typedef char BYTE;
2 typedef unsigned short USHORT;
3 typedef int MATRIX[20][40];
4 typedef int WORD;

6 main()
7 {
8     BYTE input;
9     WORD buf[512];
10    MATRIX prev, current;
11
12    ...
Referencing Structure Members
Using the Dot Operator

structure-name.member-name

```c
#include <stdio.h>
struct emp {
    char    name[21];
    char    id[8];
    double  salary;
};

main() {
    struct emp prgmr;
    char    buf[257];
    double  atof();
    ...
    gets(prgmr.name);
    gets(prgmr.id);
    gets(buf);
    prgmr.salary = atof(buf);
    printf(" Name: %s\n", prgmr.name);
    printf(" Id: %s\n", prgmr.id);
    printf("Salary: %.2f\n", prgmr.salary);
    ...
    prgmr.salary *= 1.15;    /* 15% raise */
    ...
}
```
Structures and Functions

- A structure may be passed as a function argument.
- A function may return a structure.

```c
#include <stdio.h>
#include "emp.h"
int main (void)
{
    struct emp prgmr; raise (struct emp, double);
    ...
    printf("Old salary: $%.0f\n", prgmr.salary);
    prgmr = raise(prgmr, .12);
    printf("New Salary: $%.0f\n", prgmr.salary);
    return 0;
}

struct emp raise(struct emp person, double increase)
{
    person.salary *= (1+increase);
    return(person);
}
```

**Output:**

Old Salary: $3100
New Salary: $3472
Arrays of Structures

/* Prints total of employee salaries */
#include <stdio.h>
#define NUM_EMPS 100

struct emp {
    char name[21];
    char id[8];
    double salary;
};

void fillarray (struct emp [], int);

main()
{
    struct emp staff[NUM_EMPS];
    int i;
    double sal_tot = 0;

    fillarray(staff, NUM_EMPS);
    for (i = 0; i < NUM_EMPS; i++)
        sal_tot += staff[i].salary;
    printf("Total of salaries: $%.2f\n", sal_tot);
}

staff[0]  staff[1] ...  staff[99]

<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>id</td>
<td>salary</td>
</tr>
<tr>
<td>name</td>
<td>id</td>
<td>salary</td>
</tr>
</tbody>
</table>
POINTERS TO STRUCTURES

DECLARATION FORMAT:

```c
struct tag {
  (member-decl-list)
  *pointer;
}

-OR-

struct tag *pointer;
```
POINTERS TO STRUCTURES (Contd)

```c
gstruct emp 
{ 
    char name[20];
    char id[10];
    float salary;
    *eptr, prgmr;
}
```

REFERENCE FORMAT:

`pointer -> member`

EXAMPLE:

```c
eptr = &prgmr
eptr -> name[10]
eptr -> salary
```
### Declarations and assignments

```c
struct student  temp, *p = &temp;

temp.grade = 'A';
temp.last_name = "Bushker";
temp.student_id = 590017;
```

<table>
<thead>
<tr>
<th>Expression</th>
<th>Equivalent expression</th>
<th>Conceptual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp.grade</td>
<td>p -&gt; grade</td>
<td>A</td>
</tr>
<tr>
<td>temp.last_name</td>
<td>p -&gt; last_name</td>
<td>Bushker</td>
</tr>
<tr>
<td>temp.student_id</td>
<td>p -&gt; student_id</td>
<td>590017</td>
</tr>
<tr>
<td>(*p).student_id</td>
<td>p -&gt; student_id</td>
<td>590017</td>
</tr>
</tbody>
</table>