structs in C

structs allow a programmer to group different types of data into a single entity.

```
struct persontype
  {
   char name[20];
   char rank;
   int serno;
  };
```

The above statement is the declaration of a new type!
It does NOT declare any storage!

The individual components of the struct are called members

Style point: Use the word "type" in the struct name

A struct Example

To declare a struct "variable":

```
persontype soldier;
```

To refer to members of the struct:

```
cin >> soldier.name >> soldier.rank
    >> soldier.serno;
```

`soldier` *is called an "instance" of the* `persontype` *struct*

structs with Array Members

Members can be any type, including arrays and other structs.

```
struct studentype
  {
   int studid;
   int tests[3];
   int quizzes[8];
   int final;
   char coursegrade;
  };
```


structs with Arrays

Some examples using arrays within structs

```
studentype student;

for (i = 0; i < 8; i++)
   student.quizzes[i] = 0;

testavg = (student.tests[0] +
           student.tests[1]
           student.tests[2])/3.0;
```

or

```
testavg = 0;
for (j = 0; j < 3; j++)
   testavg += student.tests[j];
testavg = testavg/3.0;
```

Arrays of structs

```
studentype allstudents[50];


for (i = 0; i < 50; i++)
   for (j = 0; j < 8; j++)
      allstudents[i].quizzes[j] = 0;


for (i = 0; i < 50; i++)
  {
   cin >> allstudents[i].studid;
   for(j = 0; j < 3; j++)
      cin >> allstudents[i].tests[j];
   for(j = 0; j < 8; j++)
      cin >> allstudents[i].quizzes[j];
   cin >> allstudents[i].final;
  }
```


structs and Functions

```
struct examptype
  {
   int a, b, c;
   float x, y;
  };

int func1(examptype);
void func2(examptype&);

int main
  {
   examptype arg;
   int r;

   r = func1(arg);
   func2(arg);
  } // END main
```

```
int func1(examptype s)
  {
   int sumints;

   sumints = s.a + s.b + s.c;
   return sumints;
  } // END func1


void func2(examptype& s)
  {
   s.x = s.a + s.b + s.c;
   s.y = s.x / 3.;
  } // END func2
```

structs are passed by value by default