Chapter 6: Structures

- A structure is a collection of one or more variables grouped together under a single name for convenient handling.
    - The variables may be of different types.
- Structures help organize complicated data.

6.1 Basics of Structures.

- Syntax of declaring a structure.

```
struct name {
    variable declarations;
};
```

- The keyword struct introduces a structure declaration.
    - A list of declarations in closed braces.
    - Name is optional and can be used as shorthand for the struct.
- Variables in a structure are called members.
    - same member names may exist in different structures w/o conflict.
- A struct declaration defines a type.
- A struct declaration that is not followed by a list of variables reserves no storage.
    - once initialized it has storage.
- An automatic structure may also be initialized by assignment or by calling a function that returns the correct structure type.

6.2 Structures and Functions.

- The only legal operations on a structure are copying it or assigning it as

a unit, taking its address with &, and accessing its members.
    - Copying and assignment include passing args to a function and returning values from functions as well.
- Structures may not be compared.

6.3 Arrays of Structures.

- Instead of making parallel arrays they can be combined in a structure.

6.4 Pointers to Structures

- if p is a pointer to a structure then arithmetic on p takes into account the size of the structure.
    - so p++ will increment to the next element.
- Do not assume that the size of a structure is the sum of the sizes of its members.
    - There may be holes in the structure data due to alignment requirements.
    - use sizeof to return the proper value.

6.5 Self-referential Structures.

- We can use a structure called a binary tree to determine occurrence of objects.
    - such as words in a file.
    - This is quicker than linear search.

6.6 Table Lookup

6.7 Typedef

- C provides a facility called typedef for creating new data types.
  - `typdef int Length;` makes Length another name for int.
  - Length can be used in declarations, casts, etc.
  - Can be used exactly the same way that int is used.
- Syntactically typedef is like the storage classes extern, static, etc.
- Can be used with structs.
- There are two main reasons for using typdefs
  - parameterize a program against portability problems.
    - If the typedefs are machine dependent they can be changed as necessary.
  - Provide better documentation for a program.

6.8 Unions

- A union is a variable that may hold objects of different types and sizes.
  - The compiler keeps track of size and alignment requirements.
- Unions provide a way to manipulate different kinds of data in a single area of storage.
  - Does not embed any machine-dependent information in the program.
- A union is a structure in which all members have offset zero from the base.
  - The structure is big enough to hold the widest member.
  - A union may only be initialized with a value of the type of its first member.
  - The same operations are permitted on unions as on structures.

6.9 Bit-fields

- When storage is low several objects can be packed into a single machine word.

    - An example is a set of single-bit flags in applications.
- A bit-field or field is a set of adjacent bits within a single implementation defined storage unit called a word.
- The syntax of definition and access is based on structures.
- Fields behave like small integers.
    - may be used in arithmetic expressions.
-