Structures and Unions in *C*

Charles Abzug, Ph.D.
Department of Computer Science
James Madison University
Harrisonburg, VA 22807

Voice Phone: ***540-568-8746;*** Cell Phone: ***443-956-9424***
E-mail: ***abzugcx@JMU.edu*** OR ***CharlesAbzug@ACM.org***
Home Page: ***http://www.cs.jmu.edu/users/abzugcx***

What Is a Structure?

1. A collection of variables that are functionally related to each other.

2. Each variable that is a member of the structure has a specific type.

3. Different members of the structure may have either the same or different types. Cf. the *elements* of an *array,* which must all be of one type.

4. A structure is a *derived* data type, constructed from two or more objects of one or more individual types.

5. The entire structure may bear a name.

6. Each member of the structure must [also] have a name.

7. The scope of the *name of a structure member* is limited to the structure itself and also to any variable declared to be of the structure's type.

(continued)

What Is a Structure? (continued)

8. THEREFORE, different structures may contain members having the same name; these may be of the same or of different types.

9. *A self-referential* structure contains a member which is a pointer to the same structure type.

10. Declaration of the structure merely defines the new data type; space is NOT reserved in memory as a result of the declaration.

    However, declaration of the structure *does* define how much memory is needed to store each variable subsequently declared to be of the type of the defined structure.

Form of Structure Declaration: Alternative 1

(1) Complete definition including assignment of a tag name to the structure.

(2) The tag name is referred to in subsequent declarations of variables of the type so defined.

(3) Each such declaration MUST include the keyword *struct* AND the name of the user-defined structure type AND the variable name(s).

```
struct nameOfThisStructureType
{
        typeOfFirstMember           nameOfFirstMember;
        typeOfSecondMember          nameOfSecondMember;
        typeOfThirdMember           nameOfThirdMember;
        .   .   .
};

struct nameOfThisStructureType  variable1OfThisStructureType,
                                variable2OfThisStructureType,
                                .   .   . ;
```

Additional variable declarations <u>can</u> subsequently be made for this structure type.

Form of Structure Declaration: Alternative 2

(1) Basic named definition of the structure is effected same as for Alternative 1.

(2) In ADDITION, one or more variables can be declared within the declaration of the structure type to be of the defined type.

(3) Other variables may also be declared subsequently to be of the same type of this structure, using the keyword *struct* together with the tag name and the variable names.

```
struct nameOfThisStructureType
{
        typeOfFirstMember          nameOfFirstMember;
        typeOfSecondMember         nameOfSecondMember;
        typeOfThirdMember          nameOfThirdMember;
        .   .   .
} variable1OfThisStructureType, variable2OfThisStructureType,  .   .   .;


struct nameOfThisStructureType  variable3OfThisStructureType,
                                variable4OfThisStructureType,
                                .   .   . ;
```

Form of Structure Declaration: Alternative 3

(1) Tag name is not assigned to the structure when the type is declared.

(2) Variables are specified within the structure declaration to be of the defined structure type.

(3) Because of the absence of a tag name for the structure type, there is no means available to ever be able to declare any other variables to be of this same type.

```
struct  /*  NO NAME ASSIGNED TO THE TYPE  */
{
        typeOfFirstMember           nameOfFirstMember;
        typeOfSecondMember          nameOfSecondMember;
        typeOfThirdMember           nameOfThirdMember;
        .   .   .
} variable1OfThisStructureType, variable2OfThisStructureType,  .   .   .;
```

Form of Structure Declaration: Alternative 4

(1) Complete definition of the structure, including assignment to it of a tag name.

(2) Subsequently, the tag name is used in a *typedef* declaration to assign a second name (i.e., an alias) to the structure. The alias can then be used in declaring a variable the same way as a native *C* type name is used, that is, <u>without</u> the keyword *struct*, i.e., just like *int, char, float,* etc.

```
struct nameOfThisStructureType
{
        typeOfFirstMember           nameOfFirstMember;
        typeOfSecondMember          nameOfSecondMember;
        typeOfThirdMember           nameOfThirdMember;
        .   .   .
};

typedef struct nameOfThisStructureType  AliasForThisStructureType;

AliasForThisStructureType  variable1OfThisStructureType,
                           variable2OfThisStructureType,   .   .   . ;
```

Form of Structure Declaration: Alternative 5

(1) Complete definition of the structure *without* assignment of a tag name.

(2) The keyword *typedef* is used *within* the declaration of the structure to assign a name (i.e., an alias) to the structure. The structure itself is anonymous, and has only the alias name. The alias can be used in the same way as a native *C* type name is used , that is, without the keyword *struct*, i.e., just like *int, char, float,* etc.

```
typedef struct
{
        typeOfFirstMember          nameOfFirstMember;
        typeOfSecondMember         nameOfSecondMember;
        typeOfThirdMember          nameOfThirdMember;
        .   .   .
}  AliasForThisStructureType;


AliasForThisStructureType  variable1OfThisStructureType,
                           variable2OfThisStructureType,   .   .   . ;
```

Example 1

```
enum genders {MALE, FEMALE};
enum studentStatus {FRESHMAN, SOPHOMORE, JUNIOR, SENIOR, POSTBAC};

struct student
{
        char firstName[20];
        char lastName[20];
        char middleName[20];
        long int studentNumber;
        short int entranceYear;
        genders studentGender;
        studentStatus status;
        char major[6];
        struct student *nextStudent;        /* Useful for making a linked list.  */
        struct student *priorStudent;       /* Useful for a doubly linked list.  */
};

struct student undergraduateStudent, graduateStudent;
struct student specialStudent;
```

Example 2

```
enum genders {MALE, FEMALE};
enum studentStatus {FRESHMAN, SOPHOMORE, JUNIOR, SENIOR, POSTBAC};

struct student
{
    char firstName[20];
    char lastName[20];
    char middleName[20];
    long int studentNumber;
    short int entranceYear;
    genders studentGender;
    studentStatus status;
    char major[6];
    struct student *nextStudent;
    struct student *priorStudent;

}   undergraduateStudent, graduateStudent;

struct student  specialStudent;
```

Example 3

```
enum genders {MALE, FEMALE};
enum studentStatus {FRESHMAN, SOPHOMORE, JUNIOR, SENIOR, POSTBAC};

struct
{
    char firstName[20];
    char lastName[20];
    char middleName[20];
    long int studentNumber;
    short int entranceYear;
    genders studentGender;
    studentStatus status;
    char major[6];
    struct student *nextStudent;
    struct student *priorStudent;

}   undergraduateStudent, graduateStudent, specialStudent;
```

Example 4

```
enum genders {MALE, FEMALE};
enum studentStatus {FRESHMAN, SOPHOMORE, JUNIOR, SENIOR, POSTBAC};

struct student
{
        char firstName[20];
        char lastName[20];
        char middleName[20];
        long int studentNumber;
        short int entranceYear;
        genders studentGender;
        studentStatus status;
        char major[6];
        struct student *nextStudent;
        struct student *priorStudent;

}   undergraduateStudent, graduateStudent;

typedef struct student StudentType;
StudentType  specialStudent;
```

Example 5

```
enum genders {MALE, FEMALE};
enum studentStatus {FRESHMAN, SOPHOMORE, JUNIOR, SENIOR, POSTBAC};

typedef struct
{
        char firstName[20];
        char lastName[20];
        char middleName[20];
        long int studentNumber;
        short int entranceYear;
        genders studentGender;
        studentStatus status;
        char major[6];
        struct student *nextStudent;
        struct student *priorStudent;

}   StudentType;

StudentType  undergraduateStudent, graduateStudent, specialStudent;
```

Which Alternative(s) Should YOU Use?

1. Alternative **3** is useful (example **3**) because it forces all variables to be declared at structure definition time.

2. Alternative **5** is useful (example **5**) because it enables variable declarations to be made to the structure type withOUT use of the keyword *struct.*

3. NONE of the other alternatives should ever be used; they are principally of historical interest.

Accessing Members of a Variable of a Structure Type

1. Structure *Member* operator ≡ *Dot* operator

```
StudentType undergraduateStudent;
char lastNameOfStudent[20];
lastNameOfStudent = undergraduateStudent.lastName;
```

1. Structure *Pointer* operator ≡ *Member* operator

```
StudentType *pointerToGraduateStudent;
short int yearOfStudentEntrance;
yearOfStudentEntrance = pointerToGraduateStudent—>entranceYear;
```

OR

```
StudentType *pointerToGraduateStudent
short int yearOfStudentEntrance;
yearOfStudentEntrance = (*pointerToGraduateStudent).entranceYear;
/* NOTE: The parentheses are NECESSARY in this example. */
```

What Is a Union?

1. Like a structure, a union is also a derived data type.

2. The members of a union share a *single* storage space.

3. Only ONE member of each union can be referenced at a time.

4. Amount of space allocated for storage is the amount needed for the *largest* member of the union.

Example of the Use of a Union

```
union temperature
{
        short int surfaceOfEarthTemperature;
        long int astronomicalTemperature;
        float floatingPointTemperature;
};

union temperature celsiusTemperature, fahrenheitTemperature, ovenTemperature,
                    surfaceOfTheSunTemperature;


main()
{
celsiusTemperature.floatingPointTemperature  =  87.3;
fahrenheitTemperature.floatingPointTemperature  =
                    32.0 + (9.0 * celsiusTemperature.floatingPointTemperature/5.0);
ovenTemperature.ssurfaceOfEarthTemperature  =  375;
surfaceOfTheSunTemperature.astronomicalTemperature  =  4387912;
}
```

END