Structures

 With array, we can only declare one data type per array.

 For different data type, we need another array declaration.

 struct overcomes this problem by declaring composite data types which can consist different types.

 A structure is a collection of related data items stored in one place and can be referenced by more than one names.

 These data items are different basic data types. So, the number of bytes required to store them may also vary.

 A structure type is a user-defined composite type.

 It is composed of fields or members which can be different types.

 In order to use a structure, we must first declare a structure template.

 The variables in a structure are called elements or members.

 In C, you must explicitly use the struct keyword to declare a

structure.

 You have the option of declaring variables when the structure type is defined by placing one or more comma-separated variable names between the closing brace and the semicolon.

struct student {

char chIdNum[5];

char chName[10];

char chGender;

int nAge;

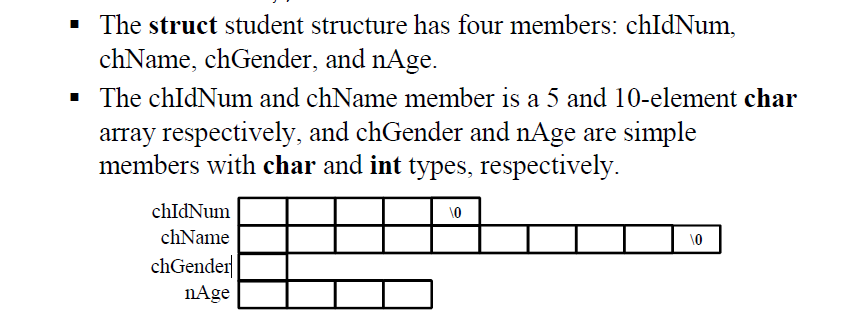
};

 Here, struct is a keyword that tells the compiler that a structure template is being declared and student is a tag that identifies its data structure.

 Tag is not a variable; it is a label for the structure’s template.

 Note that there is a semicolon after the closing curly brace.

 A structure tag simply a label for the structure’s template but you name the structure tag using the same rules for naming variables.



 Compiler will not reserve memory for a structure until it is declared.

 A structure declaration names a type and specifies a sequence of variable values called 'members' or 'fields' of the structure that can have different types.

 A variable of that structure type holds the entire sequence defined by that type. Declaring structure variables can be done in the following way,

struct student {

char chIdNum[11];

char chName[30];

char chGender;

int nAge;

}stdData;

 Another example, anonymous struct (struct without tag),

struct {

float fwidth, flength;

}area;

 The area structure has two members with float type, fwidth and flength.

 The structure type has no tag and is therefore is unnamed or anonymous.

 The anonymous structures allows you to declare a structure variable within another structure without giving it a name.

 You can access the members of an anonymous structure as If they were members in the containing structure.

 The member access operators . (dot) and -> (arrow: minus + greater than symbols) are used to refer to members of structures.

expression.identifier

 Expression represents structure variable and the identifier represents structure members.

expression->identifier

 Expression represents structure pointer and the identifier represents structure members.

(\*expression).identifier

 Alternative syntax when using structure pointer

