자료구조론 CC343_2207

Reading assignment 5

경기대학교 컴퓨터공학부

201511837 이상민

Review Questions

1. What is the advantage of using structures?
구조물을 사용하는 것의 장점은 무엇인가?

Structures gather multiple pieces of data about the same subject together in the same place. This is especially useful if one wants to gather the same few parameters (say, first name, last name, age, and phone #) about a large number of objects (say, humans). Then you could put those in a struct:

```
typedef struct PersonalData
{
   char FirstName[100];
   char LastName[100];
   int  Age;
   int  PhoneNumber;
} PersonalDataStruct;
PersonalDataStruct DataArray[1000];
// ...code here to load data into array...
```

Then you can pass pointers to members of that array to functions. Without the struct, you'd need to pass many pointers (one for each data field) to a function instead of just one.

A calling function could then make a function call like this:

```
CallOnPhone(&DataArray[i]);
```

Then the called function can access members of the struct with the "->" operator:

```
int CallOnPhone(PersonalDataStruct *dp)
{
   int ph = dp->PhoneNumber;
   // ...code here to dial phone number...
   return result; // did call succeed?
}
```

2. Structure declaration reserves memory for the structure. Comment on this statement with valid justifications.
구조 선언은 구조에 대한 기억을 남겨둔다. 이 진술에 대해 타당한 정당성을 가지고 논평하라.

How Structures are stored in Memory

Members of a structure are always stored in consecutive memory locations but the memory occupied by each member may vary. Consider the following program:

```c
#include<stdio.h>

struct book
{
    char title[5];
    int year;
    double price;
};

int main()
{
    struct book b1 = {"Book1", 1988, 4.51};

    printf("Address of title = %u\n", b1.title);
    printf("Address of year = %u\n", &b1.year);
    printf("Address of price = %u\n", &b1.price);

    printf("Size of b1 = %d\n", sizeof(b1));

    // signal to operating system program ran fine
    return 0;
}
```

Expected Output:

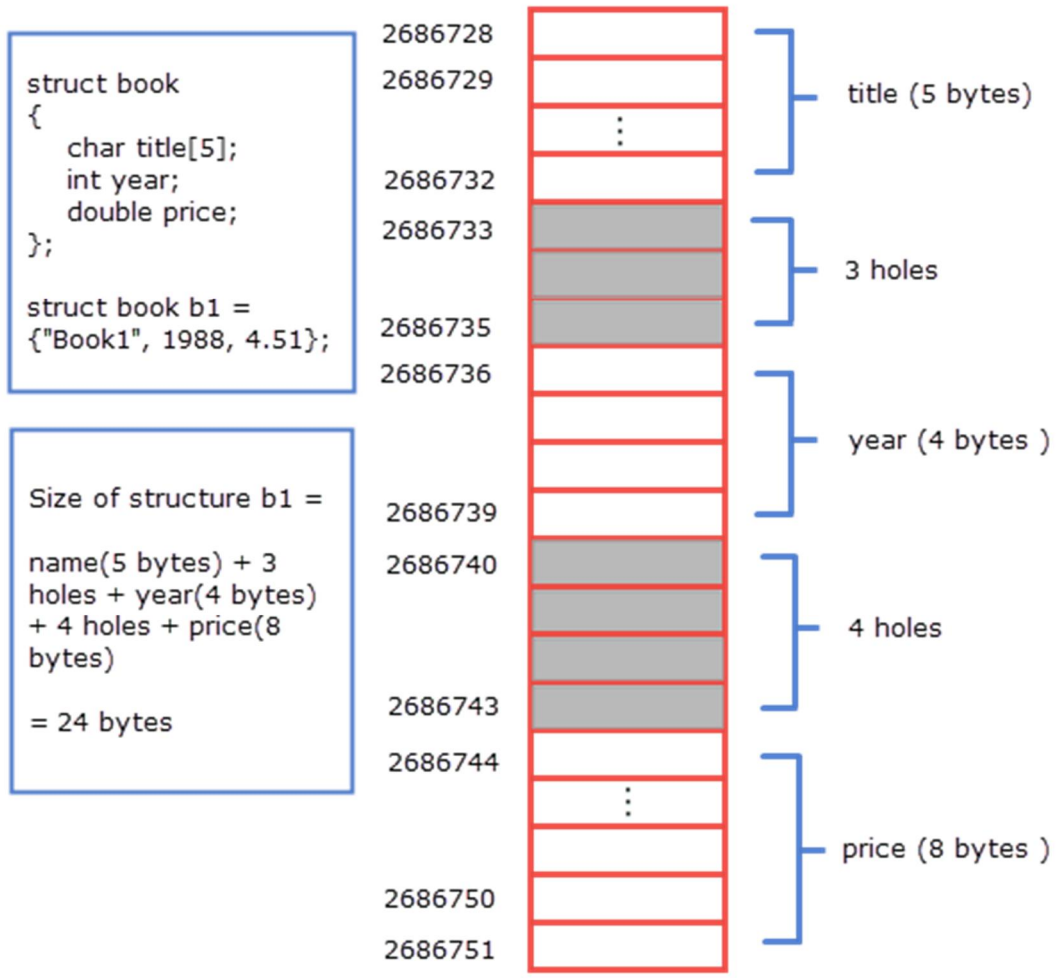
```
Address of title = 2686728
Address of year = 2686736
Address of price = 2686744
Size of b1 = 24
```

In the structure Book title occupies 5 bytes, year occupies 4 bytes and price occupies 8 bytes. So the size of structure variable should be 17 bytes. But, as you can see in the output the size of the variable b1 is 24 bytes, not 17 bytes. Why it so ?

This happens because some systems require the address of certain data types to be a multiple of

2, 4, or 8. For example, some machines store integers only at even addresses, unsigned long int and double at addresses which are multiple of 4 and so on. In our case the address of the name member is 2686728, since it is 5 bytes long, it occupies all addresses from 2686728–2686732.

The machine in which I am running these sample program stores integer numbers at multiple of 4, that's why the three consecutive bytes (i.e 2686733, 2686734, 2686735) after 2686732 are left unused. These unused bytes are called **holes**. It is important to note these holes do not belong to any member of the structure, but they do contribute to the overall size of the structure. So the next member year is stored at 2686736 (which is a multiple of 4). It occupies address 4 bytes starting from 2686736 to 2686739. Again, the next four bytes after 2686739, are left unused and eventually price member is stored at address 2686744 (which is a multiple of 8).



A structure variable in memory

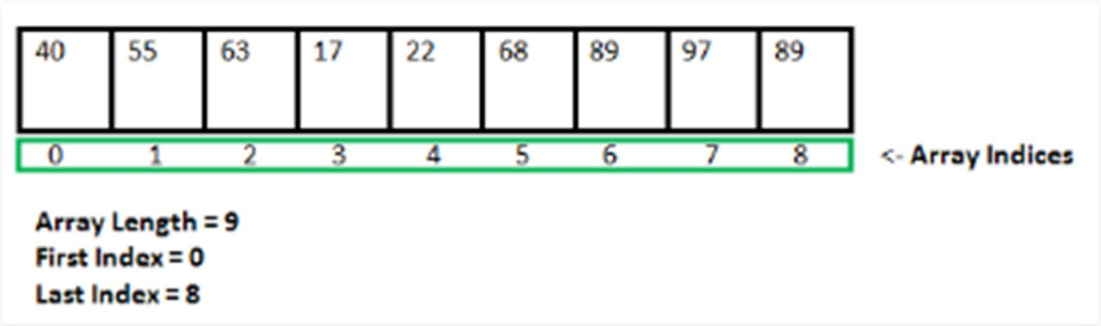
TheCguru.com

3. Differentiate between a structure and an array.

구조와 배열의 구별

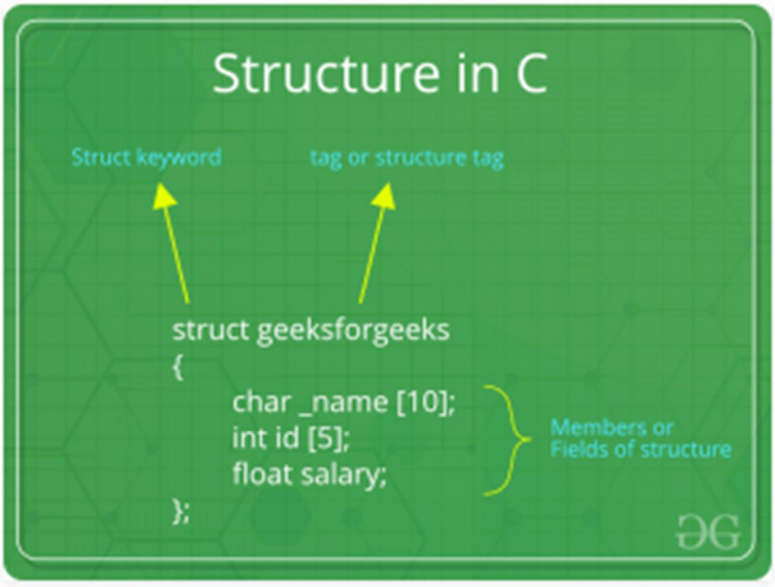
Array in C

An array is collection of items stored at continuous memory locations.



Structure in C

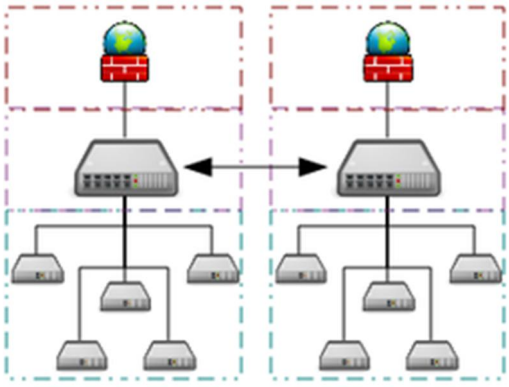
A structure is a user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.



Difference between Structure and Array

ARRAY	STRUCTURE
Array refers to a collection consisting of elements of homogenous data type.	Structure refers to a collection consisting of elements of heterogenous data type.
Array uses subscripts or "[ ]" (square bracket) for element access	Structure uses "." (Dot operator) for element access
Array is pointer as it points to the first element of the collection.	Structure is not a pointer
Instantiation of Array objects is not possible.	Instantiation of Structure objects is possible.
Array size is fixed and is basically the number of elements multiplied by the size of an element.	Structure size is not fixed as each element of Structure can be of different type and size.
Bit filed is not possible in an Array.	Bit filed is possible in an Structure.
Array declaration is done simply using [] and not any keyword.	Structure declaration is done with the help of "struct" keyword.
Arrays is a primitive datatype	Structure is a user-defined datatype.
Array traversal and searching is easy and fast.	Structure traversal and searching is complex and slow.
data_type array_name[size];	struct sruct_name(data_type1 ele1; data_type2 ele2; };
Array elements are stored in continuous memory locations.	Structure elements may or may not be stored in a continuous memory location.
Array elements are accessed by their index number using subscripts.	Structure elements are accessed by their names using dot operator.

4. Write a short note on structures and inter-process communication.
구조 및 프로세스 간 통신에 대해 짧은 메모를 작성한다.



A grid computing system that connects many personal computers over the Internet via inter-process network communication

In computer science, inter-process communication or interprocess communication (IPC) refers specifically to the mechanisms an operating system provides to allow the processes to manage shared data. Typically, applications can use IPC, categorized as clients and servers, where the client

requests data and the server responds to client requests.[1] Many applications are both clients and servers, as commonly seen in distributed computing.

IPC is very important to the design process for microkernels and nanokernels, which reduce the number of functionalities provided by the kernel. Those functionalities are then obtained by communicating with servers via IPC, leading to a large increase in communication when compared to a regular monolithic kernel. IPC interfaces generally encompass variable analytic framework structures. These processes ensure compatibility between the multi-vector protocols upon which IPC models rely.[2]

An IPC mechanism is either synchronous or asynchronous. Synchronization primitives may be used to have synchronous behavior with an asynchronous IPC mechanism.

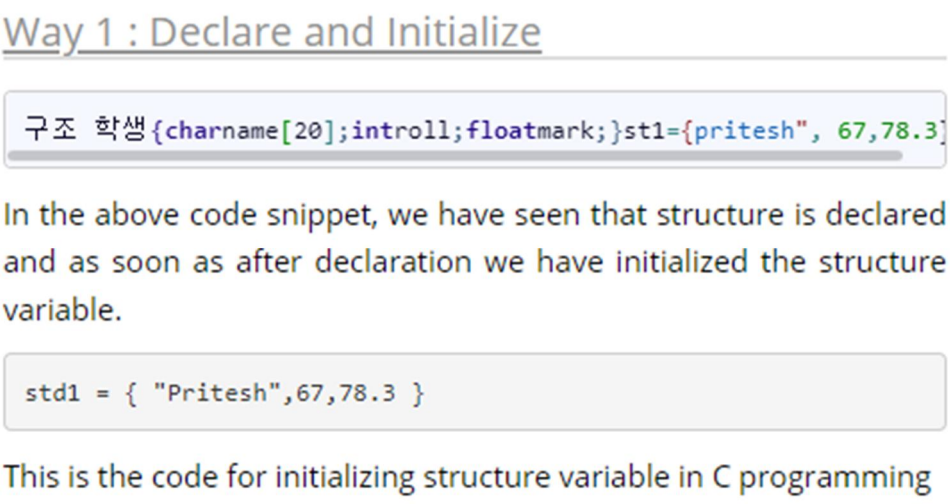
5. Explain the utility of the keyword typedef in structures.
구조물에 입력된 키워드의 효용성에 대해 설명하라.

The typedef keyword allows the programmer to create new names for types such as int or, more commonly in C, templated types--it literally stands for "type definition". Typedefs can be used both to provide more clarity to your code and to make it easier to make changes to the underlying data types that you use.

6. Explain with an example how structures are initialized.
구조가 어떻게 초기화되는지 예를 들어 설명하라.

C Structure Initialization

1. When we declare a structure, memory is not allocated for un-initialized variable.
2. Let us discuss very familiar example of structure student , we can initialize structure variable in different ways –



Way 2 : Declaring and Initializing Multiple Variables

```
struct student
{
   char name[20];
   int roll;
   float marks;
}

std1 = {"Pritesh",67,78.3};
std2 = {"Don",62,71.3};
```

In this example, we have declared two structure variables in above code. After declaration of variable we have initialized two variable.

```
std1 = {"Pritesh",67,78.3};
std2 = {"Don",62,71.3};
```

Way 3 : Initializing Single member

```
struct student
{
   int mark1;
   int mark2;
   int mark3;
} sub1={67};
```

Though there are three members of structure,only one is initialized , Then remaining two members are initialized with Zero. If there are variables of other data type then their initial values will be –

Data Type	Default value if not initialized
`integer`	0
`float`	0.00
`char`	NULL

Way 4 : Initializing inside main

```
struct student
{
    int mark1;
    int mark2;
    int mark3;
};

void main()
{
struct student s1 = {89,54,65};
- - - - --
- - - - --
- - - - --
};
```

When we declare a structure then memory won't be allocated for the structure. i.e only writing below declaration statement will never allocate memory

```
struct student
{
    int mark1;
    int mark2;
    int mark3;
};
```

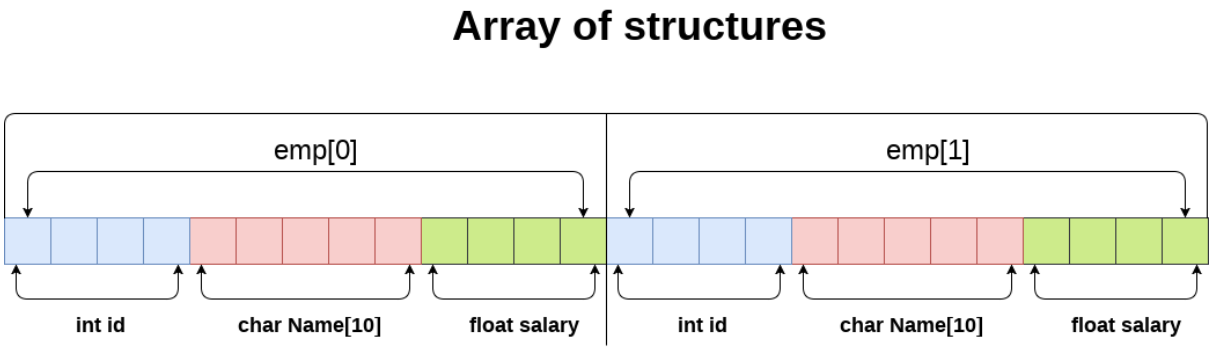
We need to initialize structure variable to allocate some memory to the structure.

```
struct student s1 = {89,54,65};
```

7. Is it possible to create an array of structures? Explain with the help of an example.
구조물의 배열을 만드는 것이 가능한가? 예를 들어 설명하시오.

Array of Structures in C

An array of structres in C can be defined as the collection of multiple structures variables where each variable contains information about different entities. The array of structures in C are used to store information about multiple entities of different data types. The array of structures is also known as the collection of structures.




```
struct employee
{
    int id;
    char name[5];
    float salary;
};
struct employee emp[2];
```

sizeof (emp) = 4 + 5 + 4 = 13 bytes

sizeof (emp[2]) = 26 bytes

```
#include<stdio.h>

#include <string.h>

struct student{

int rollno;

char name[10];

};

int main(){

int i;

struct student st[5];

printf("Enter Records of 5 students");
```

```
for(i=0;i<5;i++){
printf("₩nEnter Rollno:");
scanf("%d",&st[i].rollno);
printf("₩nEnter Name:");
scanf("%s",&st[i].name);
}
printf("₩nStudent Information List:");
for(i=0;i<5;i++){
printf("₩nRollno:%d, Name:%s",st[i].rollno,st[i].name);
}
    return 0;
}
```

Output:

```
Enter Records of 5 students
Enter Rollno:1
Enter Name:Sonoo
Enter Rollno:2
Enter Name:Ratan
Enter Rollno:3
Enter Name:Vimal
Enter Rollno:4
Enter Name:James
Enter Rollno:5
Enter Name:Sarfraz

Student Information List:
Rollno:1, Name:Sonoo
Rollno:2, Name:Ratan
Rollno:3, Name:Vimal
Rollno:4, Name:James
Rollno:5, Name:Sarfraz
```

8. What do you understand by a union?
조합이 이해한 것은?

What is a union and why does it matter?

A Union is a group of people working together to improve their work lives through collective bargaining.

What difference would a Union make?

Having a Union means that you can collectively meet and negotiate with management over any issues that affect you and your job, including wages, benefits, and working conditions. A Union contract is a legally binding document where these agreements are put in writing. Having a Union gives you a stronger voice in working with management to make the company stronger, more prosperous, and better place to work.

Who runs the Union?

You do. The Union is a democracy at every level. You elect your negotiating committee and leadership for the local, district, and international union. Our Union is made up of thousands of people like you -- standing together to make a difference.

Aren't there already laws that protect us?

Laws governing employment, safety, discrimination, and overtime all exist because millions of Union members fought for them. A Union grievance process ensures that everyone is treated fairly and equally, without favoritism and discrimination. A Union contract, with the aid of Union resources and staff, ensures that these laws and rules are enforced.

9. Differentiate between a structure and a union.
구조와 조합을 구분한다.

structures in C

A structure is a user-defined data type available in C that allows to combining data items of different kinds. Structures are used to represent a record.

Defining a structure: To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than or equal to one member. The format of the struct statement is as follows:

```
struct [structure name]
{
    member definition;
    member definition;
    ...
    member definition;
};
```

union

A union is a special data type available in C that allows storing different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple purposes.

Defining a Union: To define a union, you must use the union statement in the same way as you did while defining a structure. The union statement defines a new data type with more than one member for your program. The format of the union statement is as follows:

```
union [union name]
{
   member definition;
   member definition;
   ...
   member definition;
};
```

Similarities between Structure and Union

1. Both are user-defined data types used to store data of different types as a single unit.
2. Their members can be objects of any type, including other structures and unions or arrays. A member can also consist of a bit field.
3. Both structures and unions support only assignment = and sizeof operators. The two structures or unions in the assignment must have the same members and member types.
4. A structure or a union can be passed by value to functions and returned by value by functions. The argument must have the same type as the function parameter. A structure or union is passed by value just like a scalar variable as a corresponding parameter.
5. '.' operator is used for accessing members.

	STRUCTURE	UNION
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of **union is equal to the size of largest member.**
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

```
// C program to illustrate differences
// between structure and Union
#include <stdio.h>
#include <string.h>

// declaring structure
struct struct_example
{
        int integer;
```

```
        float decimal;
        char name[20];
};

// declaraing union

union union_example
{
        int integer;
        float decimal;
        char name[20];
};

void main()
{
        // creating variable for structure
        // and initializing values difference
        // six
        struct struct_example s={18,38,"geeksforgeeks"};

        // creating variable for union
        // and initializing values
        union union_example u={18,38,"geeksforgeeks"};


        printf("structure data:₩n integer: %d₩n"
```

```
                                "decimal: %.2f₩n name: %s₩n",
                                s.integer, s.decimal, s.name);
printf("₩nunion data:₩n integeer: %d₩n"
                                "decimal: %.2f₩n name: %s₩n",
                                u.integer, u.decimal, u.name);


// difference two and three
printf("₩nsizeof structure : %d₩n", sizeof(s));
printf("sizeof union : %d₩n", sizeof(u));

// difference five
printf("₩n Accessing all members at a time:");
s.integer = 183;
s.decimal = 90;
strcpy(s.name, "geeksforgeeks");

printf("structure data:₩n integer: %d₩n "
                                "decimal: %.2f₩n name: %s₩n",
                        s.integer, s.decimal, s.name);

u.integer = 183;
u.decimal = 90;
strcpy(u.name, "geeksforgeeks");

printf("₩nunion data:₩n integeer: %d₩n "
```

```
                                "decimal: %.2f₩n name: %s₩n",
                        u.integer, u.decimal, u.name);

printf("₩n Accessing one member at time:");

printf("₩nstructure data:");
s.integer = 240;
printf("₩ninteger: %d", s.integer);

s.decimal = 120;
printf("₩ndecimal: %f", s.decimal);

strcpy(s.name, "C programming");
printf("₩nname: %s₩n", s.name);

printf("₩n union data:");
u.integer = 240;
printf("₩ninteger: %d", u.integer);

u.decimal = 120;
printf("₩ndecimal: %f", u.decimal);

strcpy(u.name, "C programming");
printf("₩nname: %s₩n", u.name);

//difference four
```

```
        printf("₩nAltering a member value:₩n");
        s.integer = 1218;
        printf("structure data:₩n integer: %d₩n "
                                        " decimal: %.2f₩n name: %s₩n",
                                        s.integer, s.decimal, s.name);

        u.integer = 1218;
        printf("union data:₩n integer: %d₩n"
                    " decimal: %.2f₩n name: %s₩n",
                            u.integer, u.decimal, u.name);
}
```

Output:

```
structure data:
 integer: 18
 decimal: 38.00
 name: geeksforgeeks

union data:
 integeer: 18
 decimal: 0.00
 name: ?

sizeof structure: 28
sizeof union: 20

 Accessing all members at a time: structure data:
 integer: 183
 decimal: 90.00
 name: geeksforgeeks

union data:
 integeer: 1801807207
 decimal: 277322871721159510000000000.00
 name: geeksforgeeks

 Accessing one member at a time:
structure data:
integer: 240
decimal: 120.000000
name: C programming

 union data:
integer: 240
decimal: 120.000000
name: C programming

Altering a member value:
structure data:
 integer: 1218
 decimal: 120.00
 name: C programming
union data:
 integer: 1218
 decimal: 0.00
 name: ?
```

10. How is a structure name different from a structure variable?
구조 이름과 구조 변수는 어떻게 다른가?

구조체 변수를 선언할 때마다 struct 키워드를 붙여주었습니다. 하지만 이 키워드를 생략하는 방법또한 존재합니다. 이때는 typedef 키워드를 사용하여 구조체를 정의하면 됩니다. 그리고 typedef 키워드는 구조체 뿐만아니라 자료형의 별칭을 만드는 기능으로 모든 자료형의 별칭을 지정할 수 있습니다.

typedef struct 구조체이름 {
자료형 멤버이름;
} 구조체별칭;

정의를 내리는 부분에서 구조체이름과 구조체별칭은 중복되도 문제가 없습니다. 그리고 관례상 구조체이름은 앞에 _(언더슬래시)를 붙여줍니다.

typedef를 이용하여 구조체의 별칭을 만들었다면 변수를 선언할때는 더이상 struct 키워드를 사용하지않습니다. 직접 코드를 보고 typedef를 사용하였을 경우의 차이를 확인하도록 하겠습니다.

```c
#include <stdio.h>
#include <string.h>

typedef struct _Person {      // 구조체 이름은 _Person
    char name[20];            // 구조체 멤버 1
    int age;                  // 구조체 멤버 2
    char address[100];        // 구조체 멤버 3
} Person;                     // typedef를 사용하여 구조체 별칭을 Person으로 정의

int main(void)
{
    Person p1;    // 구조체 별칭 Person으로 변수 선언

    strcpy(p1.name, "박소영");
    p1.age = 22;
    strcpy(p1.address, "드림타워");

    // 점으로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1.name);       // 이름: 박소영
    printf("나이: %d\n", p1.age);        // 나이: 22
    printf("주소: %s\n", p1.address);    // 주소: 드림타워

    return 0;
}
```

구조체를 정의할 때 typedef 키워드를 사용하였고 정의를 마치는 중괄호와 세미콜론 사이에 구조체의 별칭을 지정하였습니다. Person이라는 이름으로! 밑으로 내려와 main함수를 확인하면 이전과는 다르게 struct 키워드를 명시하지 않고 Person p1;을 선언하였습니다.

```
Person p1;    // 구조체 별칭 Person으로 변수 선언
```

여기까지 typedef를 이용하여 구조체를 조금더 편리하게 사용할 수 있는 방법을 알아보았습니다. 이번에는 자료형의 별칭을 만드는 기능을 알아보도록 하겠습니다.

```
typedef 자료형 별칭
typedef 자료형* 별칭
```

```
typedef int element;  // int를 별칭 element로 정의
typedef int* Pnum;    // int 포인터를 별칭 Pnum으로 정의

element node;         // element로 변수 선언
Pnum NPtr;            // Pnum으로 포인터 변수 선언

NPtr = &node;         // 포인터에 변수의 주소를 저장
```

11. Explain how members of a union are accessed.
조합원 접근 방법 설명

C Programming Accessing union members

While accessing union, we can have access to single data member at a time. we can access single union member using following two Operators –

1. Using DOT Operator
2. Using ARROW Operator

Accessing union members DOT operator

In order to access the member of the union we are using the dot operator. DOT operator is used inside printf and scanf statement to get/set value from/of union member location.

Syntax :

variable_name.member

consider the below union, when we declare a variable of union type then we will be accessing union members using dot operator.

union emp

{

int id;

char name[20];

}e1;

id can be Accessed by – `union_variable.member`

Syntax	Explanation
`e1.id`	Access id field of union
`e1.name`	Access name field of union

Accessing union members Arrow operator

Instead of maintaing the union variable suppose we store union at particular address then we can access the members of the union using pointer to the union and arrow operator.

union emp

{

int id;

char name[20];

}*e1;

id can be Accessed by – `union_variable->member`

Syntax	Explanation
`e1->id`	Access id field of union
`e1->name`	Access name field of union

12. Write a short note on nested structures.
중첩된 구조물에 짧은 메모를 쓴다.

C – Nested Structure

- Nested structure in C is nothing but structure within structure. One structure can be declared inside other structure as we declare structure members inside a structure.
- The structure variables can be a normal structure variable or a pointer variable to access the data. You can learn below concepts in this section.

1. Structure within structure in C using normal variable

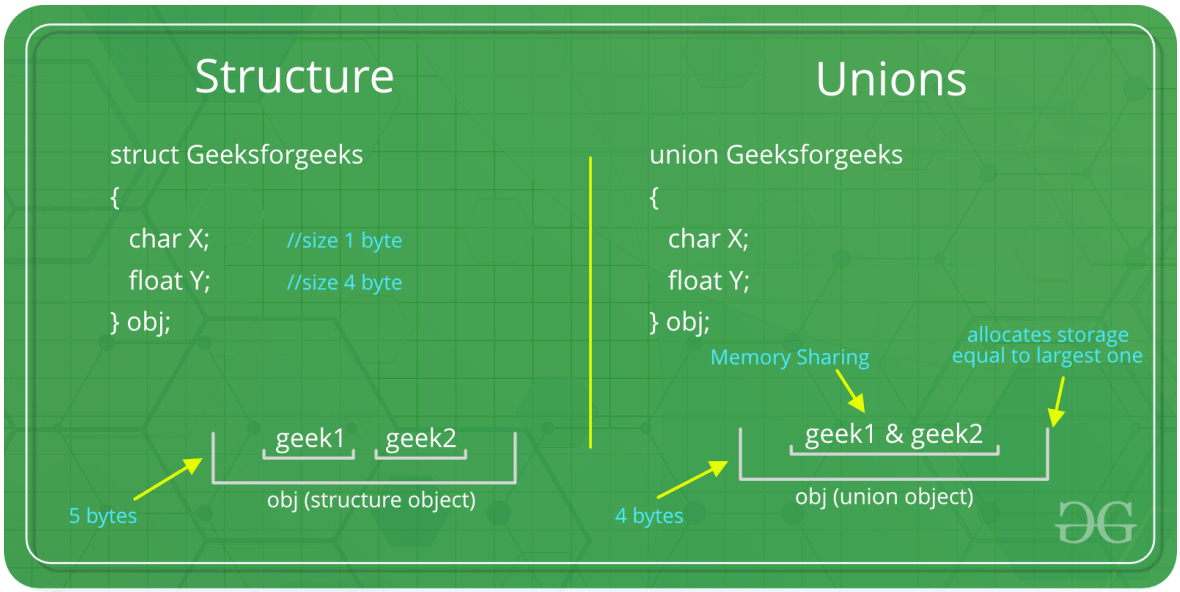
✓ This program explains how to use structure within structure in C using normal variable. "student_college_detail' structure is declared inside "student_detail" structure in this program. Both structure variables are normal structure variables.
✓ Please note that members of "student_college_detail" structure are accessed by 2 dot(.) operator and members of "student_detail" structure are accessed by single dot(.) operator.

2. Structure within structure in C using pointer variable

✓ This program explains how to use structure within structure in C using pointer variable. "student_college_detail' structure is declared inside "student_detail" structure in this program. one normal structure variable and one pointer structure variable is used in this program.
✓ Please note that combination of .(dot) and ->(arrow) operators are used to access the structure member which is declared inside the structure.

13. In which applications unions can be useful?
어떤 애플리케이션 조합이 유용할 수 있는가?

Like Structures, union is a user defined data type. In union, all members share the same memory location.



For example in the following C program, both x and y share the same location.
If we change x, we can see the changes being reflected in y.

```
#include <stdio.h>

// Declaration of union is same as structures

union test {

        int x, y;

};

int main()

{

        // A union variable t

        union test t;

        t.x = 2; // t.y also gets value 2

        printf("After making x = 2:₩n x = %d, y = %d₩n₩n",

                t.x, t.y);

        t.y = 10; // t.x is also updated to 10

        printf("After making y = 10:₩n x = %d, y = %d₩n₩n",

                t.x, t.y);

        return 0;

}
```

Output:

```
After making x = 2:
 x = 2, y = 2

After making y = 10:
 x = 10, y = 10
```

Multiple-choice Questions

1. A data structure that can store related information together is called

(a) Array (b) String (c) Structure **(d) All of these**

2. A data structure that can store related information of different data types together is called

(a) Array (b) String **(c) Structure** (d) All of these

3. Memory for a structure is allocated at the time of

(a) Structure definition

(b) Structure variable declaration

(c) Structure declaration

(d) Function declaration

4. A structure member variable is generally accessed using

(a) Address operator

(b) Dot operator

(c) Comma operator

(d) Ternary operator

5. A structure that can be placed within another structure is known as

(a) Self-referential structure **(b) Nested structure**

(c) Parallel structure (d) Pointer to structure

6. A union member variable is generally accessed using the

(a) Address operator **(b) Dot operator**

(c) Comma operator (d) Ternary operator

7. typedef can be used with which of these data types?

(a) struct (b) union (c) enum **(d) all of these**

True or False

1. Structures contain related information of the same data type. : False

2. Structure declaration reserves memory for the structure. : False

3. When the user does not explicitly initialize the structure, then C automatically does it. : True

4. The dereference operator is used to select a particular member of the structure. : True

5. A nested structure contains another structure as its member. : True

6. A struct type is a primitive data type. : False

7. C permits copying of one structure variable to another. : True

8. Unions and structures are initialized in the same way. : False

9. A structure cannot have a union as its member. : False

10. C permits nested unions. : True

11. A field in a structure can itself be a structure. : True

12. No two members of a union should have the same name. : True

13. A union can have another union as its member. : True

14. New variables can be created using the typedef keyword. : False

Fill in the Blanks

1. Structure is a _____ data type.
answer : User defined

2. _____ is just a template that will be used to reserve memory when a variable of type struct is declared.
answer : Structure declaration

3. A structure is declared using the keyword struct followed by a _____.
answer : Structure name

4. When we precede a struct name with _____, then the struct becomes a new type.
answer : Typedef

5. For int and float structure members, the values are initialized to _____.
answer : Zero

6. char and string structure members are initialized to _____ by default.
answer : Null character

7. A structure member variable is generally accessed using a _____.
answer : Dot operator

8. A structure placed within another structure is called a _____.
answer : Nested structure

9. _____ structures contain a reference to data of its same type.
answer : Self-referential

10. Memory is allocated for a structure when _____ is done.
answer : We declare a variable of a structure

11. _____ is a collection of data under one name in which memory is shared among the members.
answer : Union

12. The selection operator is used to _____.
answer : Refer to the individual members of structure or union

13. _____ permits sharing of memory among different types of data.
answer : Union