C Structures Answers

Sayak Haldar
IIEST, Shibpur

1) Answer is b) Structures

2) Answer is d) All of the mentioned

Explanation: struct, enum, typedef : all of them define user-defined data type

3) Answer is c) .

4) Answer is b) Function

5) Answer is d) None of the mentioned

6) Answer is a) Compile time error.

Explanation: New structure type definition does not end with a ';' at line 6

7) Answer is b) Compile time error.

Since, we try to initialize a member of the newly defined structure type student with in the newly defined structure type student

8) Answer is c) Compile time error

Explanation: Line 9 of the program should be replaced with struct student s

9) Answer is d) 8

10) Answer is c) Depends on the standard

11) Answer is b) Compile time error

Explanation: no is undeclared in main's scope

12) Answer is b) integer size+ character size

13) Answer is a) Compile time error.

Explanation: If the structure tag (optional field) is missing during the new structure type declaration, then the variables of the newly defined structures must be declared during the definition of the newly defined structure

1. struct //Note that: structure tag field is missing
2. {
3. int k;
4. char c;

```
}p;//so, we need to declare the variables of the newly defined structures like this
```

14) Answer is a) Compile time error

Explanation:

Compile time error due to multiple declaration of p variable. First, it is declared as a variable of an anonymous structure with members int k and char c and second is as a variable of int datatype.

15) Answer is b) 10 10

A variable name can be same as the structure tag's name

16) Answer is b) 1 a 0.000000 10

17) Answer is a) 3.000000

18) Answer is b) 0.000000

Explanation: Do not think that value 3 will be assigned to x.f

To initialize the value in such case, we have to do:

struct p x = {.c = 97, .k = 1, .f=3};

for such structure declaration:

```
struct p
{
    int k;
    char c;
    float f;
};
```

But, struct p x={97,1,3}; will work fine.

19) Answer is a) 0.000000

20) Answer is a) newton alan alan.

21) Answer is d) s[i].b

22) Answer is a) No Compile time error, generates an array of structure of size 3

23) Answer is d) All of the mentioned

24) Answer is a) void foo(struct *var)

25) Answer is b) 12

Explanation: A structure array of size 1 is created

26) Answer is Answer is a) alan alan alan turing

Explanation: Because, when we do s[1]=s[0]

the value of the members of s[0] would be copied to the members of s[1]

27) Answer is a) alan alan

28) Answer is v) Compile time error

Explanation:

Incompatible types when assigning to type ‘struct student’ from type ‘char *’

s[1] = s[0] = "alan";

29) Answer is d) 0

30) Answer is a) Compile time error

Explanation: since we cannot perform the relational operator == on structures.

That's why (p == p1) cannot be performed.

31) Answer is a) Compile time error.

Explanation: Since, we are trying to assign from struct type point to struct type notpoint. (p1=foo())

In fact, explicit typecasting is not allowed

```
#include <stdio.h>

struct point

{

        int x;

        int y;

};
```

```
struct notpoint

{

	int x;

	int y;

};

struct point foo();

int main()

{

	struct point p = {1};

	struct notpoint p1 = {2, 3};

	p1 =(struct notpoint)foo();

	printf("%d\n", p1.x);

	return 0;

}

struct point foo()

{

	struct point temp = {1, 2};

	return temp;

}
```

For, this program, compiler will tell

```
error: conversion to non-scalar type requested
p1 =(struct notpoint)foo();
```

32) Answer is a) Compile time error due to incompatible assignment.

Explanation: Compile time error due to incompatible assignment.

33) **Answer is a) Compile time error**

Explanation: We are trying to send a pointer to struct nonpoint in a function which accepts a pointer to structure point type.

The error is just the consequence of it.

34) **Answer is a) Compile time error.**

Explanation: Because, in c, ++ (post incrementation operator) has higher precedence than * operator.

That would cause the problem actually. Compiler would say:

request for member ‘x’ in something not a structure or union

```
    printf("%d\n", *p.x++);
```

However, changing that particular statement to either

```
    printf("%d\n", (*p).x++);
```

or

```
    printf("%d\n", p->x++);
```

would correct that error.

Since, '→' Member selection via pointer has higher precedence over postfix increment operator.

35) Answer is a) Compile time error.

Explanation: Now, among the operators present in the expressions, → has highest precedence. So, *p->x++

will work as *(p->x++) Now, p->x is post incremented and then p->x is replaced by the value

1. So, * operator (dereference operator) will not find any operator to work on.

36) Answer is a) Compile time error.

Explanation: Since,newly defined structure type student is not declared in the main's scope.

Notice that, we declared the structure student in the function. Declaring it globally would solve the problem.

37) Answer is b) 3

Explanation:

struct point p1[] = {1, 2, 3, 4};

By doing this, we actually create an array of struct point type with 2 elements.

38)**Answer is b) 2 2**

Explanation:

Though the actual answer should be standard dependent

The order in which the parameters to a function in not defined in the standard, and is determined by the calling convention used by the compiler. I think in this case, cdecl calling convention (which many C compilers use for x86 architecture) is used in which arguments in a function get evaluated from right to left**.**

(++p->x is performed/evaluated first)

Due to the convention followed by most of the compilers for evaluation order of function parameters.

Then p-> x will be evaluated

39) Answer is b) 2

Explanation:

++p->x

Now, → (Member selection via pointer operator) has higher precedence than ++ (pre-increment Operator) .

So, p-> x will return value 1. then the value will be incremented to 2 and the value 2 is printed.

Now, How, p-> x will return 1

p->x

=(*p).x

=p[0].x

=1

40) Answer is a) 1

p->x

=(*p).x

=p[0].x

41) Answer is b) 3 3

Though the actual answer should be standard dependent

The order in which the parameters to a function in not defined in the standard, and is determined by the calling convention used by the compiler. I think in this case, cdecl calling convention (which many C compilers use for x86 architecture) is used in which arguments in a function get evaluated from right to left**.**

printf("%d %d\n", p->y, ++p->y);

Now, ++p->y will be performed/evaluated first. (since, the evaluation order of function parameter is from right to left)

++p->y

=++((*p).y)

=++p[0].y

So, it will increment the value stored as y member of p[0] to 3

then

p->y will be printed

42) Answer is a) 1 0

Explanation:

1. struct point
2.   {
3.   int x;
4.   int y;
5.   } p[] = {1, 2, 3, 4, 5};

Here, line 5 will invoke a structure array named p with 3 members.

Now, printf("%d %d\n", p->x, p[2].y);

p->x=(*p).x=p[0].x

this will print 1

p[2].y this will print 0

43) Answer is a) 1 0

Explanation: This question is similar to the previous question. Only difference is that, in case of previous question, the array of newly defined structure with structure tag point is declared during the definition of new structure datatype. Here, in this problem, the array of newly defined structure with structure tag point is declared later.

44) Answer is b) 1 0 (at least this is the answer in my compiler)

Explanation:

Well, 1 0 is the answer at least coming in my gcc compiler.

No problem even if an array is sent as pointer to a function we could both use arr[0] and *(arr+0) for accessing the first element in the array

p->x=(*(p+0)).x=1

p[3].y=0

However, the last answer i.e. the value of p[3].y is compiler dependent. Because, in come compiler, the structure array is created with three elements whereas, in some other the structure array is created with 5 structure elements (like gcc). So, in gcc, p[3].y would have the value 0 but in some other compiler it would have garbage value.

Now, the proof that in gcc the structure array is created with five structure elements:

try to print p[4].y it would also be printed as 0

but try to print p[5].x or p[5].y they would be printed as garbage value.

45) Answer is a) Compile time error.

Explanation:

Because, we are trying to access member of y of a thing (p+2) which is neither structure nor union.

(p+2).y can be corrected to *(p+2).y to access p[2].y

46) Answer is b) 1 0

Explanation:

p->x =(*p).x =(*(p+0)).x=p[0].x (Address arithmetic. Remember?)

47) Answer is c) 16 or d) 8 depends upon the underlying machine architecture

In case of a 32 bit compiler, the size of any kind of pointer is 4.

So, in case of a 32 bit compiler, the final answer would be 8

Whereas, in case of a 64 bit compiler, the size of any kind of pointer is 8

So, in case of 64 bit compiler, the final answer would be 16

48) Answer is b) 1 or d) false depending upon the compiler

Actually, the result is compiler dependent.

In case of some compilers, struct p p1[] = {1, 92, 3, 94, 5, 96};

this assignment will invoke an array of structures with 6 elements whereas, in case of some other compilers, it invokes an array of structures with 3 elements.

If it invokes a structure with 3 elements then line 13 will be executed.

Since, x will be 5 and it is equal to sizeof(int)+sizeof(char)

Now, ptr1 will hold the structure p1[0]

i.e. by dereferencing ptr1, we will get p1[0]

So, (*ptr1).x=p1[0].x =1

Whereas, if it invokes a structure with 6 elements

then false will be printed.

49) Answer is a) compile time error

to get the result as 1, we have to change as the following:

```
#include <stdio.h>
struct p
{
    int x;
    char y;
};
typedef struct p q;
int main()
{
    struct p p1[] = {1, 92, 3, 94, 5, 96};
    q *ptr1 = p1;
    printf("%d\n", ptr1->x);
    return 0;
}
```

Or,

```
#include <stdio.h>
struct p
{
    int x;
    char y;
};
typedef struct p* q;
int main()
{
    struct p p1[] = {1, 92, 3, 94, 5, 96};
    q ptr1 = p1;
    printf("%d\n", ptr1->x);
    return 0;
}
```

But,

Not the following:

```
#include <stdio.h>
struct p
{
    int x;
    char y;
};
```

```
typedef struct p* q*;
int main()
{
    struct p p1[] = {1, 92, 3, 94, 5, 96};
    q* ptr1 = p1;
    printf("%d\n", ptr1->x);
    return 0;
}
```

This will again result compilation error.

Because, q* will not be replaced as p*. Compiler will interpret that ptr1 is a pointer of q type.

Now, q type does not exist in the program's scope. So, when -> is performed it will try to access a member x of pt1 which is neither a structure nor a union.

50) Answer is a) Compile time error

Explanation: q type is undefined in foo's scope.

51) Answer is a) *my_struct.b = 10;

Explanation: Since, '.' (Member selection via object name operator) has higher precedence over '*' (dereference operator). So, my_struct.b will be evaluated to a rvalue. And '*' operator cannot work on rvalue.

52) Answer is a) Compiler can access entire structure from the function.

Explanation: Compiler could only access the member a of the structure variable s. And it could access the whole structure in this case.

53) Answer is is d) None of the mentioned. i.e. all of them are correct.

54) Answer is a) Output will be 10.

Explanation: Since, the change made to s.a in the function change is changed locally.

55) Answer is d) false

Explanation: Now, in some compilers, structure of array would be created with 6 structure elements and in some other compilers, structure of array would be created with 3 structure elements.

Now, if structure array contains 3 structure as element x would be 24/5=4

And if structure array contains 6 structure as element x would be 48/5=9

So, answer would always be false.

56) Answer is a) hey hi

57) Answer is b) Compile time error

Explanation: This is due to incompatible assignment m.point=s

58) Answer is d) Compile time error

Explanation: Because of struct student point; this statement within in the structure declaration compilation error would be there. Compiler would say in this case:

field ‘point’ has incomplete type in struct student point.

59) Answer is c) 8 (in case of 32 bit compiler) or d) 16 (in case of 64 bit compiler)

60) Answer is a) Compile time error

Explanation: Compile time error due to sizeof(student) since student is not a identifier and it's a derived type.

i.e. we could do sizeof(int) but we could not do sizeof(student) if student is a derived datatype like enum, structure or union.

61) Answer is c) 1

62) Answer is b)1

63) Answer is b) 1

64) Answer is d) 1

65) Answer is d) All of the mentioned.

66) Answer is d) None of these

Explanation: None of these. i.e. all of these are possible.

67) Answer is b) Recursion

68) Answer is b)

69) Answer is b) i

70) Answer is c) this

71) Answer is a) Compile time error

Explanation Error: Since, p is not a pointer of structure p type. So, p->name or p->next will cause compilation error

72) Answer is d) xyz

73)Answer is b) segmentation fault/code crash

Explanation: Segmentation fault/code crash due to use of strcpy in this context. Since, memory is not allocated for q.name (which is a pointer). We first have to allocate a memory for it, then only we can use strcpy in this context

74)Answer is b) lookup(s)

75) Answer is c) Hash search

76) Answer is a) install(s,t);

77) Answer is a) install function uses lookup

78) Answer is b) It modifies the name with new definition

79) Answer is a) When there is no memory for adding new name

80) Answer is a) Compile time error

81) Answer is d) xyz