ALX LESSON

0x0E C - Structures, typedef

C - Programming

TABLE OF CONTENTS

01

OVERVIEW topics

Topics

What are structures, when, why and how to use them

How to use typedef

Slides On Telegram

https://t.me/alx_2023

C Programming

Topics

02

Learning Objectives

What are structures, when, why and how to use them

Structures

A structure is a user defined data type available in C that allows to combine data items of different kinds.

What are structures, when, why and how to use them

Structure declaration

- You can define a structure in the global scope of your program (outside of all your functions, just like the functions prototypes).

- You can declare elements of your structure in its scope.

```c
struct User
{
  char *name;
  char *email;
  int age;
};

int main(void)
{
  struct User user;

  return (0);
}
```

How to create a structure?

'struct' keyword is used to create a structure. Following is an example.

```
struct address
{
   char name[50];
   char street[100];
   char city[50];
   char state[20];
   int pin;
};
```

How to declare structure variables?

A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

```
// A variable declaration with structure declaration.
struct Point
{
int x, y;
} p1; // The variable p1 is declared with 'Point'


// A variable declaration like basic data types
struct Point
{
int x, y;
};

int main()
{
struct Point p1; // The variable p1 is declared like a normal variable
}
```

How to initialize structure members?

Structure members cannot be initialized with declaration. For example, the following C program fails in the compilation.

```
struct Point
{
int x = 0; // COMPILER ERROR: cannot initialize members here
int y = 0; // COMPILER ERROR: cannot initialize members here
};
```

The reason for above error is simple, when a datatype is declared, no memory is allocated for it. Memory is allocated only when variables are created.
Structure members can be initialized using curly braces '{}'. For example, the following is a valid initialization.

Pre defiened macros

```
struct Point
{
int x, y;
};

int main()
{
// A valid initialization. member x gets value 0 and y
// gets value 1. The order of declaration is followed.
struct Point p1 = {0, 1};
}
```

How to access structure elements?

Structure members are accessed using dot (.) operator.

```
#include <stdio.h>
struct Point {
        int x, y;
};
int main()
{
        struct Point p1 = { 0, 1 };

        // Accessing members of point p1
        p1.x = 20;
        printf("x = %d, y = %d", p1.x, p1.y);

        return 0;
}
```

What is designated Initialization?

Designated Initialization allows structure members to be initialized in any order.
This feature has been added in C99 standard.

What is designated Initialization?

```
#include <stdio.h>

struct Point {
        int x, y, z;
};

int main()
{
        // Examples of initialization using designated
        // initialization
        struct Point p1 = { .y = 0, .z = 1, .x = 2 };
        struct Point p2 = { .x = 20 };

        printf("x = %d, y = %d, z = %d\n", p1.x, p1.y, p1.z);
        printf("x = %d", p2.x);
        return 0;
}
```

What is an array of structures?

Like other primitive data types, we can create an array of structures.

```
#include <stdio.h>

struct Point {
    int x, y;
};

int main()
{
    // Create an array of structures
    struct Point arr[10];

    // Access array members
    arr[0].x = 10;
    arr[0].y = 20;

    printf("%d %d", arr[0].x, arr[0].y);
    return 0;
}
```

What is a structure pointer?

Like primitive types, we can have a pointer to a structure. If we have a pointer to structure, members are accessed using arrow (->) operator. Or (*p2).y

```
#include <stdio.h>

struct Point {
        int x, y;
};

int main()
{
        struct Point p1 = { 1, 2 };

        // p2 is a pointer to structure p1
        struct Point* p2 = &p1;

        // Accessing structure members using structure pointer
        printf("%d %d", p2->x, p2->y);
        return 0;
}
```

Limitations of C Structures

In C language, Structures provide a method for packing together data of different types. A Structure is a helpful tool to handle a group of logically related data items. However, C structures have some limitations.

The C structure does not allow the struct data type to be treated like built-in data types:
We cannot use operators like +,- etc. on Structure variables. For example, consider the following code:

Limitations of C Structures

```c
struct number {
    float x;
};
int main()
{
    struct number n1, n2, n3;
    n1.x = 4;
    n2.x = 3;
    n3 = n1 + n2;
    return 0;
}

/*Output:

prog.c: In function 'main':
prog.c:10:7: error:
invalid operands to binary + (have 'struct number' and
'struct number') n3=n1+n2;

*/
```

Limitations of C Structures

But we can use arithmetic operation on structure variables like this.

```
// Use of arithmetic operator in structure

#include <stdio.h>

struct number {
    float x;
};
int main()
{
    struct number n1, n2, n3;
    n1.x = 4;
    n2.x = 3;
    n3.x = (n1.x) + (n2.x);
    printf("\n%f", n3.x);
    return 0;
}
```

typedef

The typedef is a keyword that is used to provide existing data types with a new name. The C typedef keyword is used to redefine the name of already existing data types.

typedef

After this type definition, the identifier ‘byte’ can be used as an abbreviation for the type unsigned char, like in the example.

```c
typedef unsigned char byte;

int main(void)
{
  byte c;

  c = 200;
  return (0);
}
```

typedef

C typedef Syntax

```
typedef existing_name alias_name;
```

After this declaration, we can use the alias_name as if it were the real existing_name in out C program.

Example of typedef in C

```
typedef long long ll;
```

C program to implement typedef

```c
#include <stdio.h>

// defining an alias using typedef
typedef long long ll;

// Driver code
int main()
{
        // using typedef name to declare variable
        ll var = 20;
        printf("%ld", var);

        return 0;
}
```

typedef struct

typedef can also be used with structures in the C programming language. A new data type can be created and used to define the structure variable.

C program to implement typedef

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

// using typedef to define an alias for structure
typedef struct students {
		char name[50];
		char branch[50];
		int ID_no;
} stu;

// Driver code
int main()
{
		stu st;
		strcpy(st.name, "Kamlesh Joshi");
		strcpy(st.branch, "Computer Science And Engineering");
		st.ID_no = 108;

		printf("Name: %s\n", st.name);
		printf("Branch: %s\n", st.branch);
		printf("ID_no: %d\n", st.ID_no);
		return 0;
}
```

typedef with Pointers

typedef can also be used with pointers as it gives an alias name to the pointers. Typedef is very efficient while declaring multiple pointers in a single statement because pointers bind to the right on the simple declaration.

Example:

```
typedef int* Int_ptr;
Int_ptr var, var1, var2;
```

C program to implement typedef

```c
// typedef with pointers
#include <stdio.h>

typedef int* ptr;

// Driver code
int main()
{
        int a = 20;
        ptr var = &a;
        *var = 15;

        printf("Value of var is %d", *var);
        return 0;
}
```

04

Hands on lab Practice


Have a Question
Leave a Comment!

Share

To let the others know more

Subscribe

To stay updated with latest videos

Thanks