Structures & Typedef

Why do we need structure?

Structure

- A structure is a user defined data type that allows us to combine different data types.
- Syntax

```
struct structure_name {
        datatype1 member1;
        datatype2 member2;
};
```

Defining structure

```
struct Student
{
    char *name;
    char *email;
    float scores;
};
```

- The struct "Student" is defined with three members: name, email, and scores. The name member is of the type char pointer, email is of type char and scores is of type float.

Initializing structures

struct structure_name variable_name = {value1, value2, value3...};

struct Student student1 = {“John Doe”, “j.doe@gmail.com”, 124.3};

- This initializes a struct student object named "student1" with the values "John Doe" for the name member, “j.doe@gmail.com” for the email member and 124.3 for the scores member.

Accessing structure members in C

- We can access the values of the members of the `struct Student` object `"student1"` by using the dot operator `(.)`

  ```
  student1.name;
  student1.email;
  student1.scores;
  ```

- After accessing the members, we can then manipulate them as we wish.

Full example demo.c

```c
#include <stdio.h>

struct Student
{
        char *name;
        char *email;
        float scores;
};

int main()
{
        struct Student student1 = {"John Doe", "j.doe@gmail.com", 124.3};
        printf("\t\nName         : %s",student1.name);
        printf("\t\nEmail        : %s",student1.email);
        printf("\t\nScores       : %f",student1.scores);

        return 0;
}
```

...another way demo1.c

```c
#include <stdio.h>

struct Student
{
        char *name;
        char *email;
        float scores;
};

int main()
{
        struct Student student1;
        student1.name = "John Doe";
        student1.email = "j.doe@gmail.com";
        student1.scores = 124.34;

        printf("\t\nName        : %s",student1.name);
        printf("\t\nEmail       : %s",student1.email);
        printf("\t\nScores      : %f",student1.scores);

        return 0;
}
```

Pointers to Structures

- We can also use pointers for structures.
- To create pointer we use the same syntax we've been using along.
- We can get access to the members of a structure by dereferencing.
- A simpler way to do it is using the symbol "->"

...pointers demo2.c

```c
#include <stdio.h>

struct Student
{
        char *name;
        char *email;
        float scores;
};

int main()
{
        struct Student student1;
        struct Student *ptr;
        ptr = &student1;
        //dereferencing the pointer before accessing the data with '.' symbol

        (*ptr).name = "John Doe";

        //We can also use '->' to do the same
        ptr->email = "j.doe@gmail.com";
        ptr->scores = 124.467;

        printf("\t\nName        : %s",student1.name);
        printf("\t\nEmail       : %s",student1.email);
        printf("\t\nScores      : %f",student1.scores);

        return 0;
}
```

Write a function that creates a new structure object student demo3.c

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

struct Student
{
        char *name;
        char *email;
        float scores;
};

struct Student *new_student(char *name, char *email, float scores)
{
        struct Student *student;
        student = malloc(sizeof(struct Student));
        if (student == NULL)
                return (NULL);
        student->name = name;
        student->email = email;
        student->scores = scores;
        return student;
}
int main()
{
        struct Student *student;

        student = new_student("Jane Doe", "janedoe@gmail.com", 22.795);
        if (student == NULL)
                return (-1);
        printf("\t\nName        : %s",student->name);
        printf("\t\nEmail       : %s",student->email);
        printf("\t\nScores      : %f",student->scores);

        return 0;
}
```

Typedef

- This is a keyword we use in C to give a type a new name.
- Syntax

  typedef type new_name;

- After this type definition, the identifier '*size_t*' can be used as an abbreviation for the type unsigned int, like in the example.

  typedef unsigned int ***size_t; // used when we use sizeof***

- Thus, any time a variable is declared to be size_t, it is actually declared to be unsigned int and it will behave as an unsigned int.

- **Typedef and structures** demo4.c

```c
#include<stdio.h>

struct Student {
        char *name;
        char *email;
        float scores;
};

typedef struct Student student_t;

int main() {
        student_t student1;

        student1.name = "John Doe";
        student1.email = "j.doe@gmail.com";
        student1.scores = 123.54646;

        printf("\t\nName        : %s",student1.name);
        printf("\t\nEmail       : %s",student1.email);
        printf("\t\nScores      : %f",student1.scores);

    return 0;
}
```

...another way to use typedef demo5.c

```c
#include<stdio.h>

typedef struct Student {
        char *name;
        char *email;
        float scores;
} student_t;


int main() {
        student_t student1;

        student1.name = "John Doe";
        student1.email = "j.doe@gmail.com";
        student1.scores = 123.54646;

        printf("\t\nName        : %s",student1.name);
        printf("\t\nEmail       : %s",student1.email);
        printf("\t\nScores      : %f",student1.scores);

    return 0;
}
```

Final example demo6.c

Write a function that creates a new structure object student use typedef to rename our structure demo6.c

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

typedef struct Student
{
        char *name;
        char *email;
        float scores;
} student_t;

student_t *new_student(char *name, char *email, float scores)
{
        student_t *student;
        student = malloc(sizeof(struct Student));
        if (student == NULL)
                return (NULL);
        student->name = name;
        student->email = email;
        student->scores = scores;
        return student;
}
int main()
{
        student_t *student;

        student = new_student("Jane Doe", "janedoe@gmail.com", 22.795);
        if (student == NULL)
                return (-1);
        printf("\t\nName        : %s",student->name);
        printf("\t\nEmail       : %s",student->email);
        printf("\t\nScores      : %f",student->scores);

        return 0;
}
```

See you at the next session!