02_10_ C Typedef

Making `typedef` Definitions

- You can create your own names for data types with the help of the **`typedef`** keyword in C, and make those name synonyms for the data types.
- Then, you can use the name synonyms, instead of the data types themselves, in your programs.
- Often, the name synonyms defined by **`typedef`** can make your program more readable.

Why Use `typedef`?

- There are several advantages to using typedef definitions.
  - First, you can consolidate complex data types into a single word and then use the word in variable declarations in your program.
- In this way, you don't need to type a complex declaration over and over, which helps to avoid typing errors.
  - Second, is that you just need to update a typedef definition, which fixes every use of that typedef definition if the data type is changed in the future.
- **`typedef`** is so useful, in fact, that there is a header file called **`stddef.h`** included in the ANSI-standard C that contains a dozen typedef definitions.

C – Typedef

- **Typedef** is a keyword that is used to give a new symbolic name for the existing name in a C program.
- This is same like defining alias for the commands.
- Consider the below structure.

```
struct student
{
        int mark [2];
        char name [10];
        float average;
}
```

C – Typedef

- Variable for the above structure can be declared in two ways.
- 1st way :

```
struct student record;       /* for normal variable */
struct student *record;      /* for pointer variable */
```

- 2nd way :

```
typedef struct student status;
```

C – Typedef

- When we use "**`typedef`**" keyword before **`struct <tag_name>`** like above, after that we can simply use type definition "status" in the C program to declare structure variable.

- Now, structure variable declaration will be, "status record".

- This is equal to "struct student record".

- Type definition for "struct student" is status. i.e. status = "struct student"

An alternative way for structure declaration using typedef in C

- To declare structure variable, we can use the below statements.

```
typedef struct student
{
          int mark [2];
          char name [10];
          float average;
} status;
```

```
status record1;                    /* record 1 is structure variable */
status record2;                    /* record 2 is structure variable */
```

Structure using typedef

- This program is used to store and access "**id**, **name** and **percentage**" for one student. We can also store and access these data for many students using array of structures.

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

typedef struct student
{
  int id;
  char name[20];
  float percentage;
} status;

int main()
{
  status record;
  record.id=1;
  strcpy(record.name, "Raju");
  record.percentage = 86.5;
  printf(" Id is: %d \n", record.id);
  printf(" Name is: %s \n", record.name);
  printf(" Percentage is: %f \n", record.percentage);
  return 0;
}
```

```
Id is: 1
Name is: Raju
Percentage is: 86.500000
```

Structure using typedef

- **Typedef** can be used to simplify the real commands as per our need.
- For example, consider below statement.

```
typedef long long int LLI;
```

- In above statement, **`LLI`** is the type definition for the real C command "**`long long int`**".
- We can use type definition **`LLI`** instead of using full command "**`long long int`**" in a C program once it is defined.

Structure using typedef

```
#include <stdio.h>
#include <limits.h>

int main()
{
   typedef long long int LLI;

   printf("Storage size for long long int data type  : %ld \n", sizeof(LLI));

   return 0;
}
```

```
Storage size for long long int data type : 8
```

End of 02_21