Typedef

Its purpose is to redefine the name of an existing variable type.

```c
#include <stdio.h>
int main(){
      typedef int my_own_type;
      my_own_type x;
      x=2;
      printf("%d\n",x);

      return 0;
}

/*Output*/

2
```

Thus, typedef provides a short and meaningful way to call a data type. Usually, uppercase letters are used to make it clear that we are dealing with a renamed data type.

```c
#include <stdio.h>
int main(){
      struct employee{
            char name[10];
            int age;
      };
      struct employee E1={"Someone",30};
      printf("Name:%s and
age:%d\n",E1.name,E1.age);
      return 0;
}

/*Output*/

Name:Someone and age:30
```

=

```c
#include <stdio.h>
int main(){
      struct employee{
            char name[10];
            int age;
      };
      typedef struct employee EMPLOYE;
      EMPLOYE E1={"Someone",30};
      printf("Name:%s and
age:%d\n",E1.name,E1.age);
      return 0;
}
/*Output*/

Name:Someone and age:30
```

Thus, by reducing the length and apparent complexity of data types, typedef can help to clarify source listing and save time and energy spent in understanding a program.

Also simplifies the creation of pointer variables.

```c
#include <stdio.h>
int main(){
      struct employee{
            char name[10];
            int age;
      };
      typedef struct
employee EMPLOYE;
      EMPLOYE
E1={"Someone",30};
      EMPLOYE *ptr=&E1;
      printf("%u\n",ptr);
      return 0;
}

/*OUTPUT*/

6422016
```