

# typedef command

the C programming language provides a keyword called **typedef** to set an **alternate name** to an existing data type.

The **typedef** keyword in C is very useful in assigning a convenient alias to a built-in data type as well as any derived data type such as a struct, a union or a pointer.

Sometimes it becomes clumsy to use a data type with a longer name (such as "**struct structname**" or "**unsigned int**") every time a variable is declared. In such cases, we can assign a handy shortcut to make the code more readable.

## typedef Syntax

In general, the **typedef** keyword is used as follows –

```
typedef existing_type new_type;
```

## typedef Examples

### Example 1

In C language, the keyword "**unsigned**" is used to declare unsigned integer variables that can store only non-negative values.

C also has a keyword called "**short**" that declares an integer data type that occupies 2 bytes of memory. If you want to declare a variable that is **short** and can have only non-negative values, then you can combine both these keywords (unsigned and short):

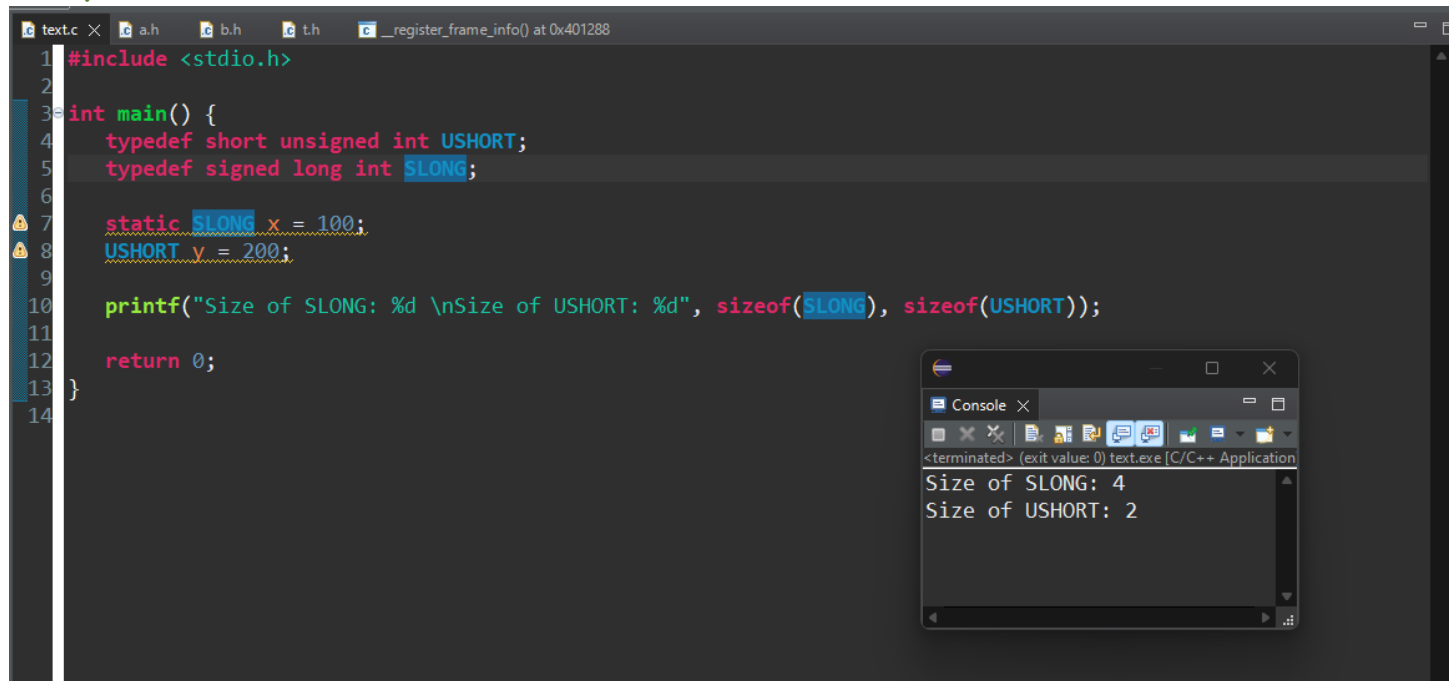
```
short unsigned int x;
```

If there are going to be many variables to be declared of this type, it will not be very convenient to use these three keywords every time. Instead, you can define an **alias** or a shortcut with the **typedef** keyword as follows –

```
typedef short unsigned int USHORT;
```

```
USHORT x;
```

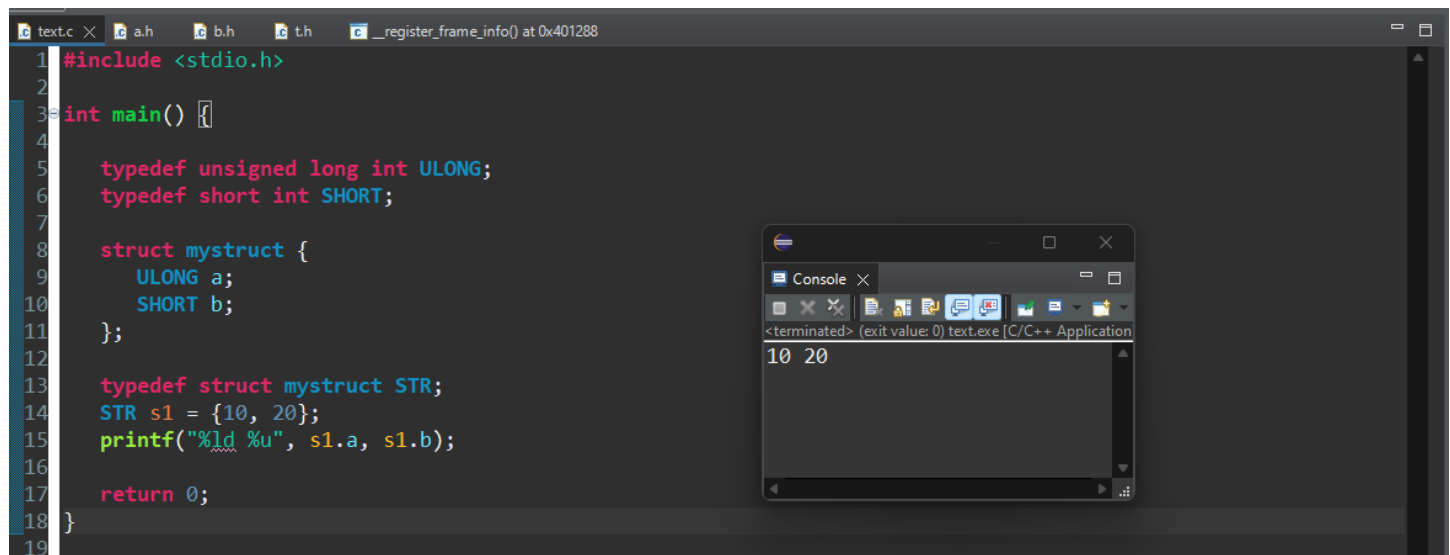
## Example 2



```
1 #include <stdio.h>
2
3 int main() {
4     typedef short unsigned int USHORT;
5     typedef signed long int SLONG;
6
7     static SLONG x = 100;
8     USHORT y = 200;
9
10    printf("Size of SLONG: %d \nSize of USHORT: %d", sizeof(SLONG), sizeof(USHORT));
11
12    return 0;
13 }
14
```

The console output shows the sizes of the typedefs: Size of SLONG: 4 and Size of USHORT: 2.

## Defining a Structure using Typedef

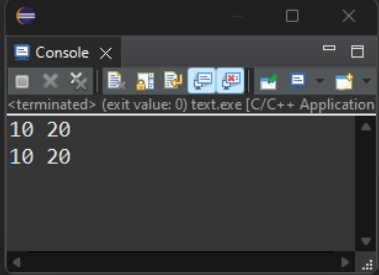


```
1 #include <stdio.h>
2
3 int main() {
4
5     typedef unsigned long int ULONG;
6     typedef short int SHORT;
7
8     struct mystruct {
9         ULONG a;
10        SHORT b;
11    };
12
13    typedef struct mystruct STR;
14    STR s1 = {10, 20};
15    printf("%ld %u", s1.a, s1.b);
16
17    return 0;
18 }
19
```

The console output shows the values of the structure: 10 20.

## Typedef for Struct Pointer

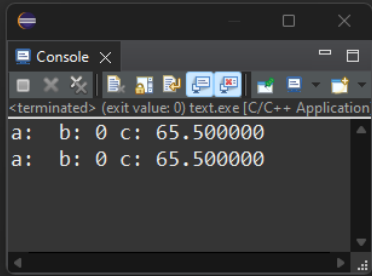
```
1 #include <stdio.h>
2
3 int main() {
4
5     typedef unsigned long int ULONG;
6     typedef short int SHORT;
7
8     typedef struct mystruct {
9         ULONG a;
10        SHORT b;
11    } STR;
12
13    STR s1 = {10, 20};
14    typedef STR * strptr;
15    strptr ptr = &s1;
16
17    printf("%d %d\n", s1.a, s1.b);
18    printf("%d %d", ptr->a, ptr->b);
19
20    return 0;
21 }
22
```



The console window shows the output of the program: 10 20 and 10 20.

## Typedef for Union

```
1 #include <stdio.h>
2
3 int main() {
4
5     typedef unsigned long int ULONG;
6     typedef short int SHORT;
7
8     typedef union myunion {
9         char a;
10        int b;
11        double c;
12    } UNTYPE;
13
14    UNTYPE u1;
15    u1.c = 65.50;
16
17    typedef UNTYPE * UNPTR;
18    UNPTR ptr = &u1;
19
20    printf("a:%c b: %d c: %lf\n", u1.a, u1.b, u1.c);
21    printf("a:%c b: %d c: %lf\n", ptr->a, ptr->b, ptr->c);
22
23    return 0;
24 }
25
```

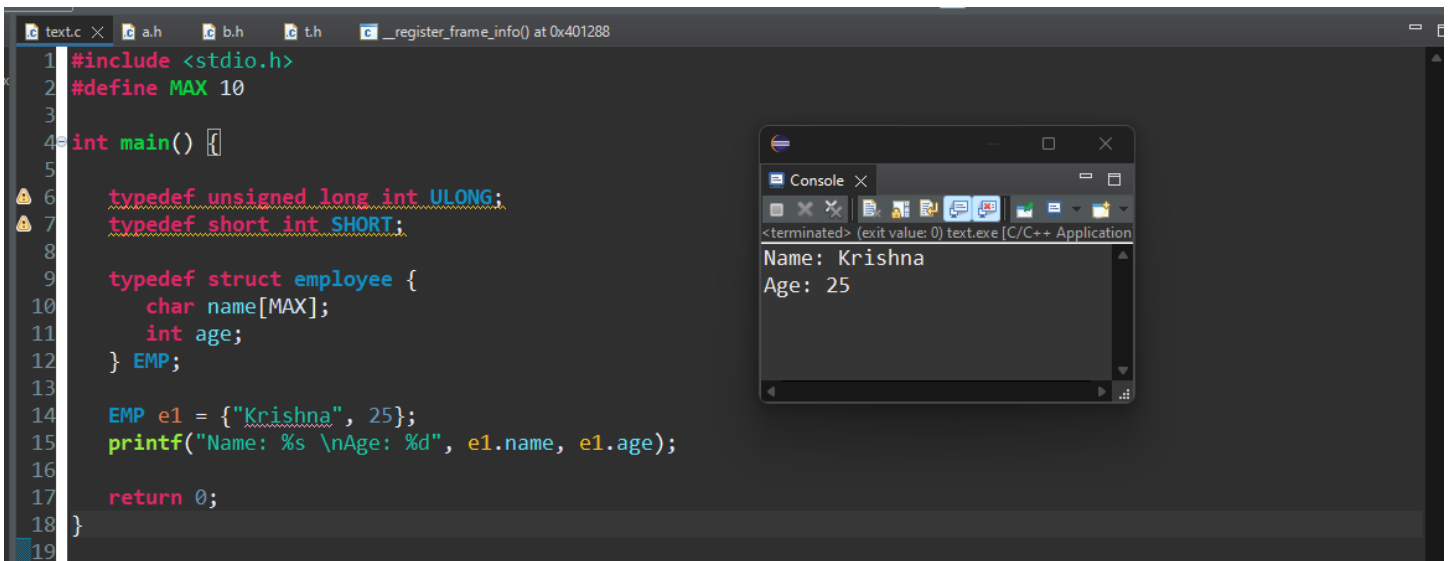


The console window shows the output of the program: a: b: 0 c: 65.500000 and a: b: 0 c: 65.500000.

## typedef vs #define in C

In C language, **#define** is a preprocessor directive. It is an effective method to define a constant. **#define** is a preprocessor directive, while **typedef** is evaluated at the time of compilation.

- **typedef** is limited to giving symbolic names to types only.
- **#define** can be used to define alias for values as well. For example, you can define "1" as "ONE".
- **typedef** interpretation is performed by the compiler.
- **#define** statements are processed by the pre-processor.



The screenshot shows a C program in a code editor with the following code:

```
1 #include <stdio.h>
2 #define MAX 10
3
4 int main() {
5
6     typedef unsigned long int ULONG;
7     typedef short int SHORT;
8
9     typedef struct employee {
10         char name[MAX];
11         int age;
12     } EMP;
13
14     EMP e1 = {"Krishna", 25};
15     printf("Name: %s \nAge: %d", e1.name, e1.age);
16
17     return 0;
18 }
```

Overlaid on the code editor is a console window titled "Console" showing the output of the program:

```
<terminated> (exit value: 0) text.exe [C/C++ Application]
Name: Krishna
Age: 25
```

