Typedef:

- The typedef keyword allows the programmer to create a new datatype name for existing datatype.
- No new datatype is produced but an alternate name is given to a known datatype.
- General format:

```
typedef<existing datatype> <new datatype, ....>
```

- typedef statement does not occpy storage, it simply defines a new type.
- typedef statements can be placed anywhere in C program as long as they come prior to their first use in the code.
- C language allows a programmer to rename datatype using keyword typedef.

Example:

```
typedef int id_number;
typedef float weight;
```

In the above example, id_number is the new datatype name given to datatype int, while weight is the new data-type name given to datatype float.

- Therefore the following statement

```
id_number vinay, komal, jaspal;

weight apples, pears, mangoes;
```

Mean that,

↳ vinay, kamal, jaspal are variable names that are declared to hold 'int' datatype.

↳ The new datatype id_number suggests that the data content of the variable names vinay, kamal and jaspal

- are integers and their identification number.

* typedef makes the code more portable.

Complex datatype like structure can use the typedef keyword

```
typedef struct point
{
  int x;
  int y;
}Dot;
 Dot left, right;
```

Shows the left and right are the structure variables of structure 'point'

* When typedef is used to name a structure the structure name tag name is not necessary.

```
typedef struct
{
  float real;
  float imaginary;
} Complex;
 complex u,v;
```

In this 'u' and 'v' as complex numbers having a real part and an imaginary part.

Example program :

```
#include<stdio.h>
typedef int myint;
int main()
{
    int x;
    myint y;
    typedef myint smallint;
    smallint z;
    printf("enter two values");
    scanf("%d %d", &x, &y);
    z=x+y;
    printf("sum value is: %d", z);
    getch();
   return 0;
}
```

output:

```
enter two values: 4, 2
sum value is: 6
```

structure padding:

- structure padding is a Concept in c that adds the one more empty bytes between the memory addresses to align the data in memory
- suppose we create a user defined structure. when we create an object of this structure, then the contigious memory will be allocated to the structure member.

```
struct student
{
  char a;
  char b;
  int c;
} stu1;
```