What happens when a function is called before its declaration in C?

In C, if a function is called before its declaration, the **compiler assumes return type of the function as int**.

For example, the following program fails in compilation.

```
#include <stdio.h>
int main(void)
{
    // Note that fun() is not declared
    printf("%d\n", fun());
    return 0;
}

char fun()
{
   return 'G';
}
```

```
prog.c: In function 'main':
prog.c:5:17: warning: implicit declaration of function 'fun'
  printf("%d\n", fun());
                 ^
prog.c: At top level:
prog.c:9:6: error: conflicting types for 'fun'
 char fun()
      ^
```

The following program compiles and run fine because return type of fun() is changed to int.

```
#include <stdio.h>
int main(void)
{
    printf("%d\n", fun());
    return 0;
}

int fun()
{
   return 10;
}
```

What about parameters? compiler assumes nothing about parameters. Therefore, the compiler will not be able to perform compile-time checking of argument types and arity when the function is applied to some arguments. This can cause problems. For example, the following program compiled fine in GCC and produced garbage value as output.

```c
#include <stdio.h>

int main (void)
{
    printf("%d", sum(10, 5));
    return 0;
}
int sum (int b, int c, int a)
{
    return (a+b+c);
}
```

There is this misconception that the compiler assumes input parameters also int. Had compiler assumed input parameters int, the above program would have failed in compilation.