Basic Library Functions for I/O

- We have already seen that the function `scanf()` takes input from the keyboard and the function `printf()` writes the output on the VDU.
- These functions are not written by us, but are supplied with the C compiler as library functions (/lib/libc-2.3.4.so)[a].

[a] There are many such functions in the C library called libc. There are other libraries as well e.g. the library of mathematical functions libm.

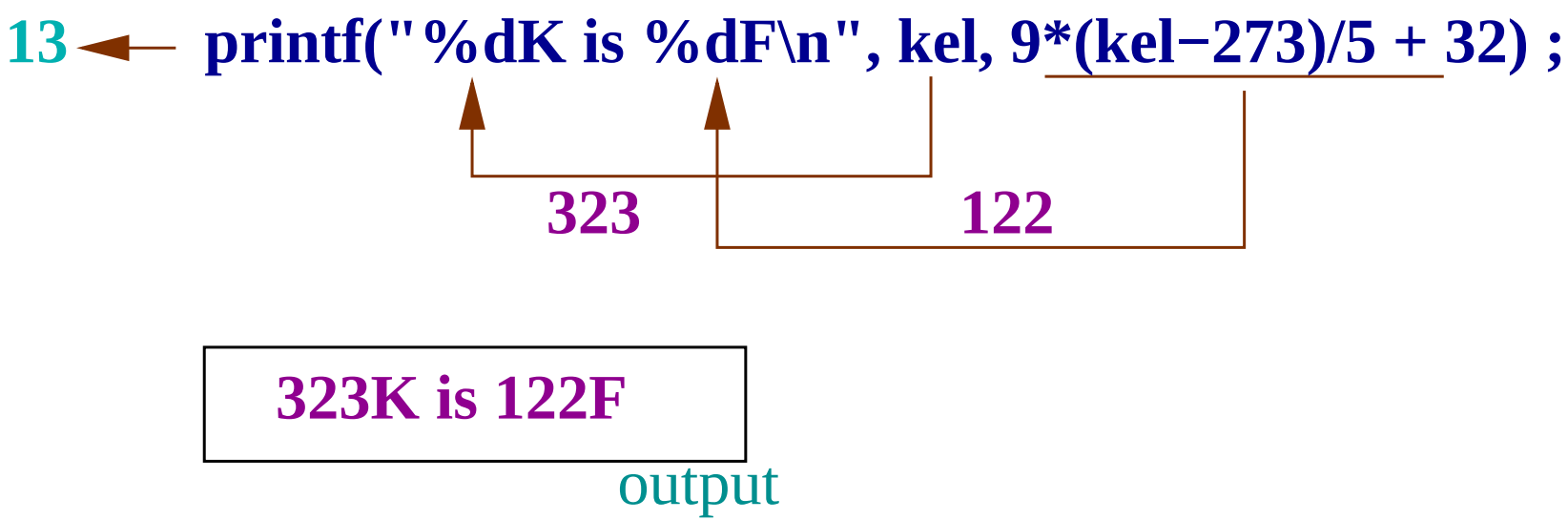
- The prototypes of `scanf()`, `printf()` and many other standard I/O functions are available in the header file `stdio.h` (/usr/include/stdio.h).
- Both these functions take variable number of arguments (at least one).
- In both the cases the first argument contains the information about the number and types of the following arguments.

```
int printf(char *format, ...)
```

- The first parameter `format`, a string of characters, has two types of objects.
  1. Ordinary characters, directly copied to the output stream.
  2. Argument specifiers e.g. %d, %c, %f, %e, %p, %lf, %u, %hd, %x etc. specify the conversion types of the following arguments[a].

[a] The specifier `%d` is for `int`, `%e` for `float` in scientific notation etc.

- The function `printf()` converts the arguments into string of characters and puts them in place of the corresponding specifiers.
- It returns the number of characters it has put in the output stream of `stdout`.


13
printf("%dK is %dF\n", kel, 9*(kel−273)/5 + 32) ;
323
122
323K is 122F
output

```
#include <stdio.h>
int main() // temp15.c
{
    int kel = 273+50, count;
    count = printf("%dK is %dF\n",
                kel, 9*(kel-273)/5+32);
    printf("No. of char: %d\n", count) ;
    return 0 ;
}
```

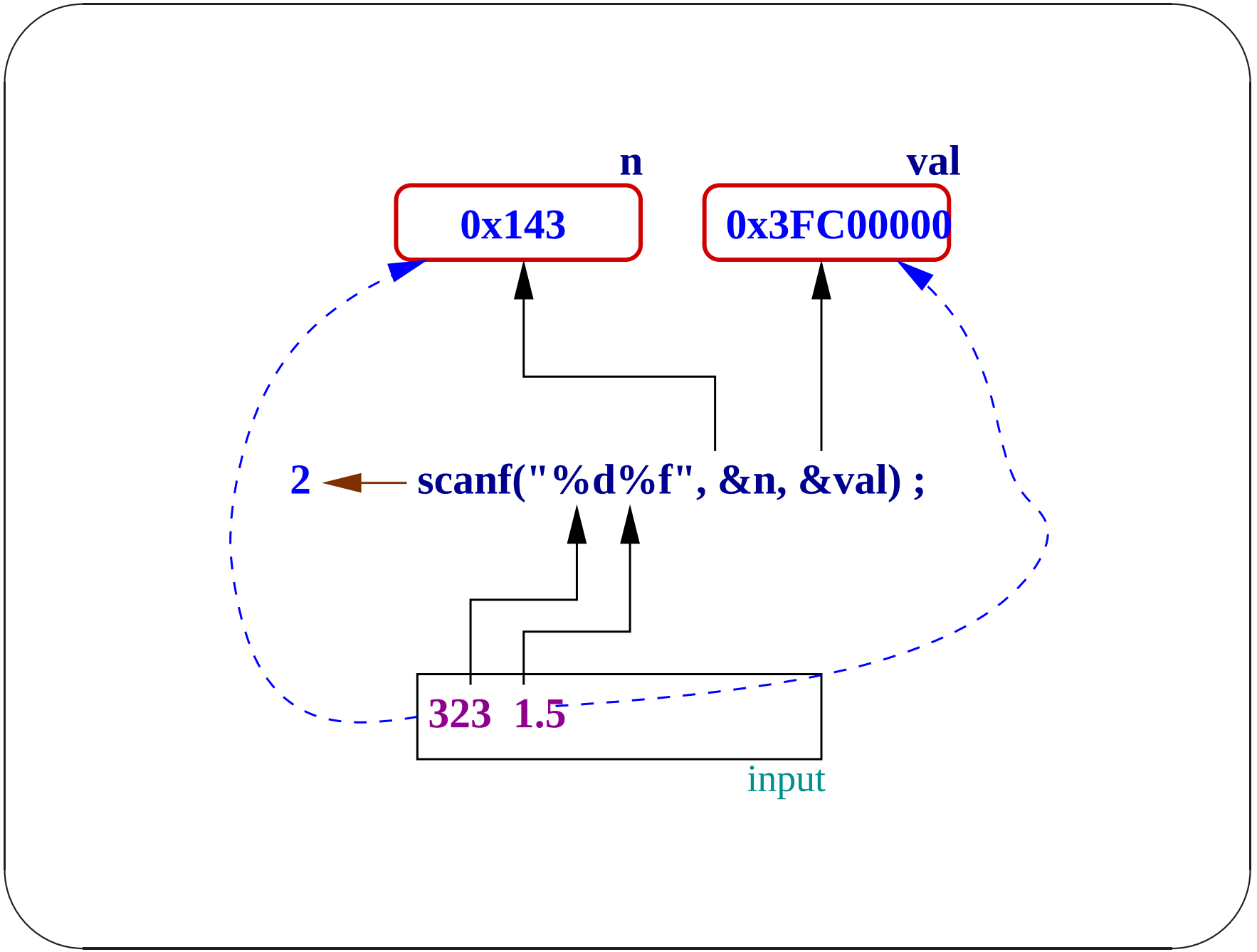
```
$ cc -Wall temp15.c
$ ./a.out
323K is 122F
No.  of char:  13
```

```
int scanf(char *format, ...)
```

- The function scanf() reads character stream from the standard input (`stdin`) device (keyboard is mapped).
- The input is converted to the internal representation following the argument specifiers in the format string.
- The corresponding values are stored in the locations specified by the arguments.

- All arguments to `scanf()` are of type pointer.
- The `scanf()` stops when all the arguments specified are read, or when there is a mismatch in the specification or the end of input (EOF) is reached[a]
- The function returns the number of data items successfully read[b].

[a] In C on Linux Ctrl+D on the keyboard is equivalent to EOF.
[b] It returns a special value on EOF (often -1).


n
val
0x143
0x3FC00000
2
scanf("%d%f", &n, &val) ;
323 1.5
input

```
#include <stdio.h>
int main() // temp16.c
{
    int n, count;
    float f ;
    count = scanf("%d%f", &n, &f) ;
    printf("No. of data: %d\n", count) ;
    return 0 ;
}
```

```
$ cc -Wall temp16.c
$ ./a.out
10 1.5
No.  of data:  2
```

```
$ ./a.out
10 <Ctrl+D>
No.  of data:  1
```

```
$ ./a.out
<Ctrl+D>
No.  of data:  -1
```

```
$ ./a.out
.5
No.  of data:  0
```

```
int getchar(void)
```

- The function `getchar()` reads a character from the standard input device (`stdin` - keyboard) and returns its ASCII value as an `unsigned char` cast to an `int`.
- It returns `EOF` at the end-of-file or error.