String & Characters in C

Difference in explicit

GeeksForGeeks - Difference between `scanf()` and `gets()` in C

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {

    // Reads the next character from stdin.
    int ch;
    /* read/print "abcde" from stdin */
    while ((ch=getchar()) != EOF)
          printf("%c", ch);

    // Reads stdin into the character array pointed to by str
    // until a newline character is found or end-of-file occurs.
    //
    // It is used to read input from the standard input(keyboard).
    // It is used to read the input until it encounters newline or End Of
File(EOF).
    char stringToBeCutted[105];
    while (gets(stringToBeCutted) != NULL) { }

    // Reads data from a variety of sources,
    // interprets it according to format and stores the results into given
location.
    //
    // It is used to read the input(character, string, numeric data) from
the standard input(keyboard).
    // It is used to read the input until it encounters a whitespace,
newline or End Of File(EOF).
    int a;
    scanf("%d", &a);
}
```

getchar

cppreference.com - getchar

Defined in header <stdio.h>

```c
int getchar(void);
```

Reads the next character from stdin.

Equivalent to `getc(stdin)`.

Parameters

None

Return value

The obtained character on success or EOF on failure.

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int ch;
    while ((ch=getchar()) != EOF)   /* read/print "abcde" from stdin */
          printf("%c", ch);
}
```

gets

cppreference.com - gets

Defined in header <stdio.h>

`char *gets( char *str );` (removed in C11)

`char *gets_s( char *str, rsize_t n);` (Since C11)

1. Reads stdin into the character array pointed to by str until a newline character is found or end-of-file occurs. A null character is written immediately after the last character read into the array. **The newline character is discarded but not stored in the buffer**.

Parameters for gets

str - character string to be written

Return value for gets

str on success, a numm pointer on failure.

Notes

The `gets()` function does not perform bounds checking, therefore this function is extremely vulnerable to buffer-overflow attacks. It cannot be used safely. For this reason, **the function has been deprecated in the third corrigendum to the C99 standard and removed altogether in the C11 standard**. `fgets()` and `gets_s()` are the recommended replacements.

Never use gets().

scanf

Defined in header <stdio.h>

`int scanf( const char *format, ...);` until C99

`int scanf( const char *restrict format, ...);` since C99

Reads data from a variety of sources, interprets it according to format and stores the results into given location.

reads the data from stdin

Parameters for scanf

- stream - input file stream to read from
- buffer - pointer to a null-terminated character string to read from
- format - pointer to a null-terminated character string specifying how to read the input
- ... - receiving arguments.

The format string consists of:

- non-whitespace multibyte characters except `%`: each such character in the format string consumes exactly one identical character from the input stream, or causes the function to fail if the next character on the stream does not compare equal.
- whitespace characters: any single whitespace character in the format string consumes all available consecutive whitespace characters from the input (determined as if by calling isspace in a loop). Note that there is no difference between `"\n"`, `" "`, `"\t\t"`, or other whitespace in the format string.
- conversion specificatitons. Each conversion specification has the following format:
  - introductory `%` character
  - (optional) assignment-suppressing character `*`.
  - (optional) integer number(greater than zero) that specifies maximum field width.
  - (optional) length modifier that specifies the size of the receiving argument, that is, the actual destination type.
  - conversion format specifier

The following format specifiers are available:

Conversion specifier	Explanation
d	matches a decimal integer.
i	matches an integer.
f	matches a floating-point number.

Notes for scanf

Because most conversion specifiers first consume all consecutive whitespace, code such as

```
scanf("%d", &a);
scanf("%d", &b);
```

will read two integers that are entered on different lines (second `%d` will consume the newline left over by the first) or on the same line, separated by spaces or tabs (second `%d` will consume the spaces or tabs).

The conversion specifiers that do not consume leading whitespace, such as `%c` can be made to do so by using a whitespace character in the format string:

```
scanf("%d", &a);
scanf(" %c", &c); // consume all consecutive whitespace after %d, then
read a char
```