ESc 101: Fundamentals of Computing

Lecture 18

Feb 11, 2010

Outline

Strings in C

- Strings are represented by an array of characters in C.
- The last element of the array is the `NULL` character, or the ASCII value 0.
- This is represented by `'\0'`.

Strings in C

- Strings are represented by an array of characters in C.
- The last element of the array is the `NULL` character, or the ASCII value 0.
- This is represented by `'\0'`.

Example

```
main()
{
    char a_string[27];

    for (int i = 0; i < 26; i++)
        a_string[i] = 'a' + i;

    a_string[26] = '\0';

    printf("The string is: %s\n", a_string);
}
```

Constant Strings

Another way of writing strings is by enclosing it in double quotes:

```
"This is a string"
```

These are called string constants.

Constant Strings

- Strings stored as arrays can be modified, but constant strings cannot.
- Strings cannot be a part of an expression in C.

Constant Strings

- Strings stored as arrays can be modified, but constant strings cannot.
- Strings cannot be a part of an expression in C.

Invalid Statements

```
char a_string[27];
char b_string[27];

a_string = "abcd";

b_string = a_string;

(a_string == b_string)
```

Invalid Statements

```
char a_string[27];
char b_string[27];

a_string = "abcd";

b_string = a_string;

(a_string == b_string)
```

Invalid Statements

```
char a_string[27];
char b_string[27];

a_string = "abcd";

b_string = a_string;

(a_string == b_string)
```

Valid Use of Strings

- Strings can be passed as argument to functions.
- If the passed string is an array, it can be modified.
- However, if it is a constant, it cannot be modified.
- `printf` and `scanf` functions accept string arguments.

Valid Use of Strings

- Strings can be passed as argument to functions.
- If the passed string is an array, it can be modified.
- However, if it is a constant, it cannot be modified.
- `printf` and `scanf` functions accept string arguments.

Valid Use of Strings

- Strings can be passed as argument to functions.
- If the passed string is an array, it can be modified.
- However, if it is a constant, it cannot be modified.
- `printf` and `scanf` functions accept string arguments.

Outline

printf

The general format of `printf` is:

```
printf( <string constant>, argument-1, ..., argument-k )
```

`printf`

- The `<string constant>` is a constant string specifying what needs to be output.
- It contains special commands, each starting with %.
- There are exactly k special commands.

`printf`

- The `<string constant>` is a constant string specifying what needs to be output.
- It contains special commands, each starting with %.
- There are exactly k special commands.

printf

Suppose `<string constant>` is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string `<s1>`,
- Output the value stored in `argument-1` treating it as integer,
- Output string `<s2>`,
- Output the value stored in `argument-2` treating it as a symbol,
- Output string `<s3>`,
- Output value stored in `argument-3` treating it as a string,
- Output string `<s4>`,
- Output value stored in `argument-4` treating it as a real number, ...

printf

Suppose `<string constant>` is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string `<s1>`,
- Output the value stored in `argument-1` treating it as integer,
- Output string `<s2>`,
- Output the value stored in `argument-2` treating it as a symbol,
- Output string `<s3>`,
- Output value stored in `argument-3` treating it as a string,
- Output string `<s4>`,
- Output value stored in `argument-4` treating it as a real number, ...

printf

Suppose <string constant> is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string <s1>,
- Output the value stored in argument-1 treating it as integer,
- Output string <s2>,
- Output the value stored in argument-2 treating it as a symbol,
- Output string <s3>,
- Output value stored in argument-3 treating it as a string,
- Output string <s4>,
- Output value stored in argument-4 treating it as a real number, ...

printf

Suppose `<string constant>` is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string `<s1>`,
- Output the value stored in `argument-1` treating it as integer,
- Output string `<s2>`,
- Output the value stored in `argument-2` treating it as a symbol,
- Output string `<s3>`,
- Output value stored in `argument-3` treating it as a string,
- Output string `<s4>`,
- Output value stored in `argument-4` treating it as a real number, ...

printf

Suppose <string constant> is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string <s1>,
- Output the value stored in argument-1 treating it as integer,
- Output string <s2>,
- Output the value stored in argument-2 treating it as a symbol,
- Output string <s3>,
- Output value stored in argument-3 treating it as a string,
- Output string <s4>,
- Output value stored in argument-4 treating it as a real number, ...

printf

Suppose `<string constant>` is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string `<s1>`,
- Output the value stored in `argument-1` treating it as integer,
- Output string `<s2>`,
- Output the value stored in `argument-2` treating it as a symbol,
- Output string `<s3>`,
- Output value stored in `argument-3` treating it as a string,
- Output string `<s4>`,
- Output value stored in `argument-4` treating it as a real number, ...

printf

Suppose `<string constant>` is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string `<s1>`,
- Output the value stored in `argument-1` treating it as integer,
- Output string `<s2>`,
- Output the value stored in `argument-2` treating it as a symbol,
- Output string `<s3>`,
- Output value stored in `argument-3` treating it as a string,
- Output string `<s4>`,
- Output value stored in `argument-4` treating it as a real number, ...

printf

Suppose `<string constant>` is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string `<s1>`,
- Output the value stored in `argument-1` treating it as integer,
- Output string `<s2>`,
- Output the value stored in `argument-2` treating it as a symbol,
- Output string `<s3>`,
- Output value stored in `argument-3` treating it as a string,
- Output string `<s4>`,
- Output value stored in `argument-4` treating it as a real number, ...

printf

Suppose `<string constant>` is:

```
"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"
```

Its meaning is:

- Output string `<s1>`,
- Output the value stored in `argument-1` treating it as integer,
- Output string `<s2>`,
- Output the value stored in `argument-2` treating it as a symbol,
- Output string `<s3>`,
- Output value stored in `argument-3` treating it as a string,
- Output string `<s4>`,
- Output value stored in `argument-4` treating it as a real number, ...