

Computer Programming

Dr. Deepak B Phatak

Dr. Supratik Chakraborty

Department of Computer Science and Engineering
IIT Bombay

Session: Handling Text Data Using 'scanf' and 'printf'

Quick Recap and overview



- We saw that normal text input and output is actually handled by C++ programs using files `stdin` and `stdout`
- We know that the standard input operator `cin` and `cout` handle conversion of data from/to ASCII characters to/from internal representation
- We will study another powerful mechanism to handle text I/O and perform such conversion

Formatted input/output



- Special functions to perform formatted input and output operations on stdin and stdout

`scanf()` and `printf()`

- Parameters to these functions include a “format” string, followed by variables/expressions to be read/printed
- C++ applies the appropriate format pattern to each value
 - for interpreting characters in the input string and converting these to internal representation
 - for generating output string from given expressions

The printf() function

- Converts values as per a specified format string

```
int roll = 12345, int batch = 112;
```

```
....
```

```
printf(“%5d %3d\n”, roll, batch)
```

This will produce the following output line on stdout

```
“12345 112\n”
```

printf(“format-string”, value, value, ...)



- This function displays one or more values on the user terminal
`printf("%d is a number\n", N);`
- If value of N is, say, 523, output produced by this function call is:
523 is a number
- This format string has a “format specifier” (%d), which is used to interpret N and convert it to a formatted value. Other characters are displayed as they are. \n introduces a new line
- ‘Specifiers’ can appear anywhere, each must correspond to a value appearing after the format string

Examples of format specifiers



`%6d` - 6 digit integer

`%7s` - string fitted in 7 characters

`%8.2f` - float, 8 digits total, 2 after decimal point

`%8.2g` - same as float, switch to E notation if required

`scanf("format-string", &var, &var, ..)`



```
int M, N; float x, y; char name[40];
```

```
scanf("%d %d %f %f %s", &M, &N, &x, &y, name);
```

- Any one of the following lines of text data will be interpreted correctly, with same values being assigned to variables

```
25  -78  .00763  345.29  Mynameschandra
```

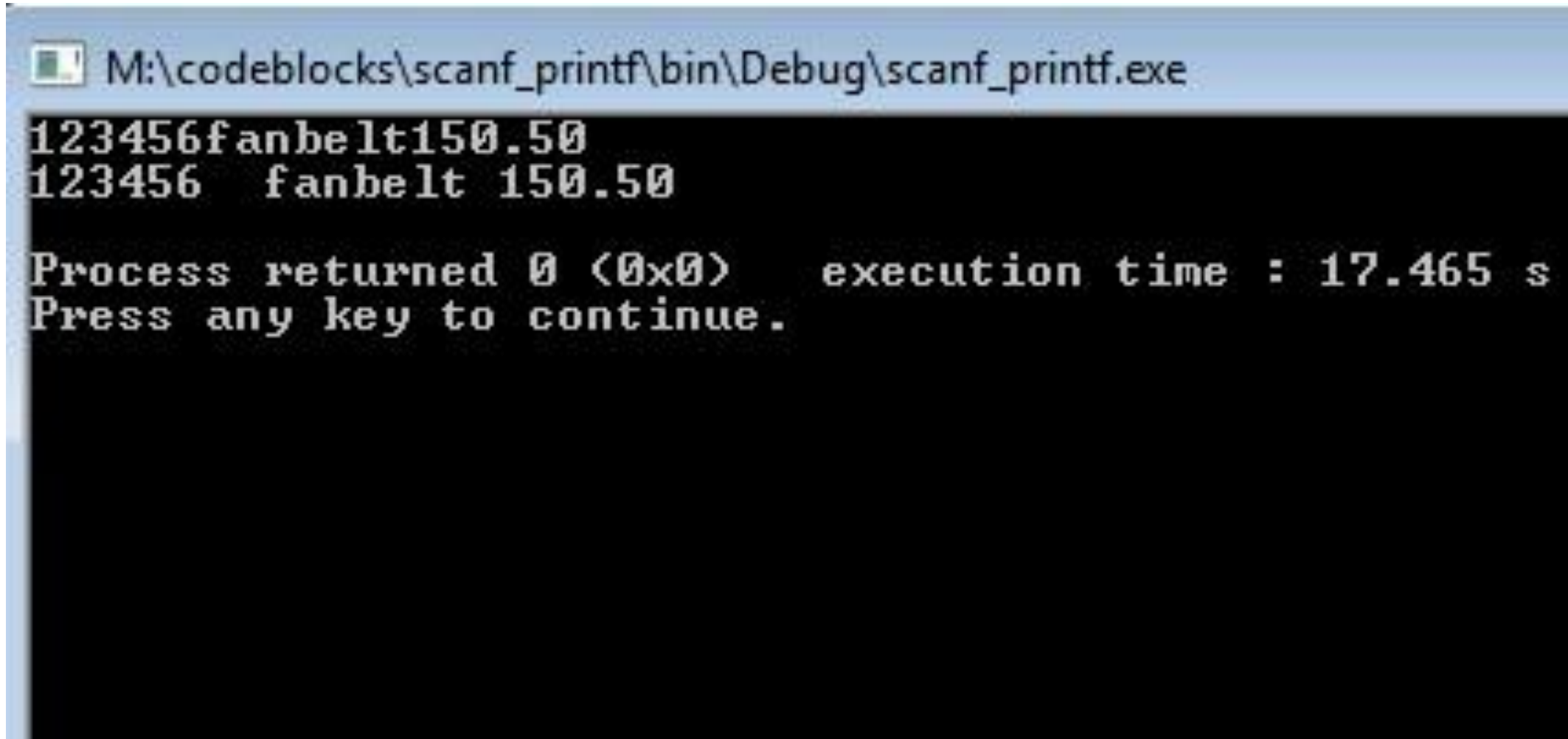
```
25    -78    7.63E-3  3.4529E2  Mynameschandra
```

Another example of scanf()



```
int a; float x; char itemcode[8];  
// The input data line contains  
// 123456fanbelt150.50  
scanf("%6d%7s%f", &a, itemcode, &x);  
printf("%6d\t%7s\t%6.2f\n", a, itemcode, x);
```


Executing scanf/printf



```
M:\codeblocks\scanf_printf\bin\Debug\scanf_printf.exe
123456fanbelt150.50
123456  fanbelt 150.50

Process returned 0 (0x0)    execution time : 17.465 s
Press any key to continue.
```

Different versions of these functions



- Interpret input values from a string/create an output string
`sprintf(s, “format string”, expression, expression, ...)`
`sscanf(s, “fomat-string”, &var1, &var2, ...)`
- Interpret input from a line, to be read from a text file
`fscanf(fpin, “fomat-string”, &var1, &var2, ...)`
- Output a formatted line to a text file
`fprintf(fpout, “format string”, expression, expression, ...)`

Summary



- We studied how to handle formatted text using functions `scanf()` and `printf()`; and their different versions
- Refer to C++ tutorials and reference section on the web at: <http://www.cplusplus.com/reference>
- Study different format specifiers