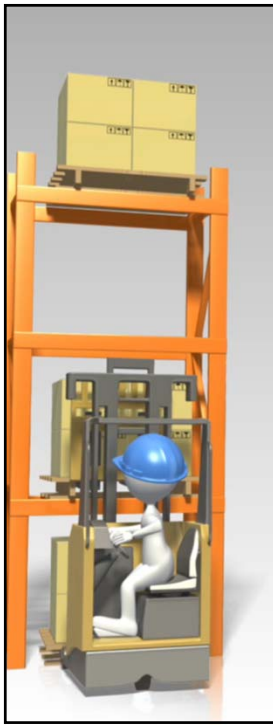
IO Stream in C Language

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
INTERNATIONAL UNIVERSITY, VIETNAM NATIONAL UNIVERSITY

Presenter:
HA VIET
UYEN SYNH,
Ph.D.



Outline

Streams

General I/O streams in C language

File I/O streams in C language

I. Streams

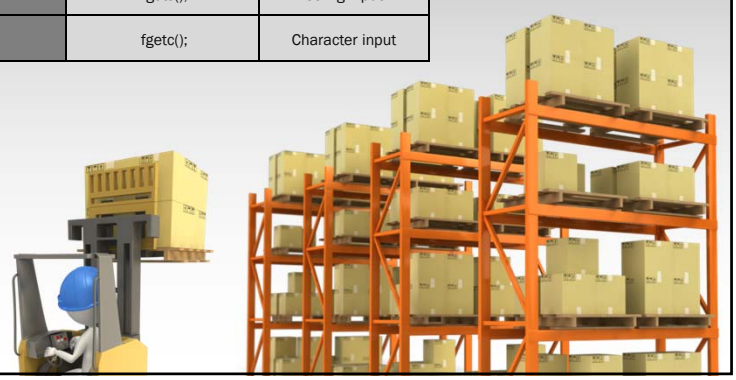
Language	Name	Stream	Device
C <stdio.h>	stdin	Standard input	Keyboard
	stdout	Standard output	Screen
C++ <iostream>	istream	Standard input	Keyboard
	ostream	Standard output	Screen

Predefined Streams

ANSI C/C++ has four predefined streams, also referred to as the *standard input/output files*

II. C's General Stream Functions

Uses One of the Standard Streams	Requires a Stream Name	Description
printf()	fprintf(); sprintf()	Formatted output
puts();	fputs()	String output
putchar(); putc();	fputc();	Character output
scanf();	fscanf(); sscanf();	Formatted input
gets();	fgets();	String input
getchar(); getc();	fgetc();	Character input



printf(); fprintf(); sprintf();

Syntax:

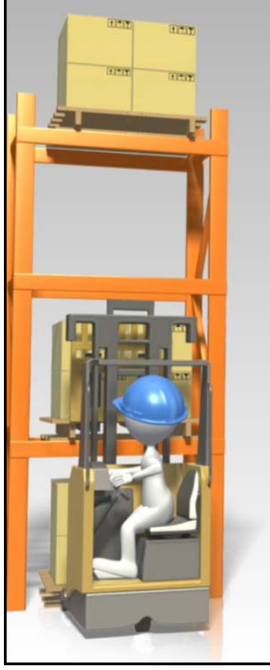
int printf(const char *format, ...); (1)

int fprintf(FILE *stream, const char *format, ...); (2)

int sprintf(char *buffer, const char *format, ...); (3)

Example:

```
Input: printf("Color %s, Number %d, Float %5.2f", "red", 123456, 3.14;)
Output: Color red, Number 123456, Float 3.14
```



Format placeholders

Syntax:

%[*flags*][*width*][*.precision*][*length*]*type*

%[-|+|(space)|#|0][(number)|][.(number)|*]*
[h|i|L]type

Flags: (optional)

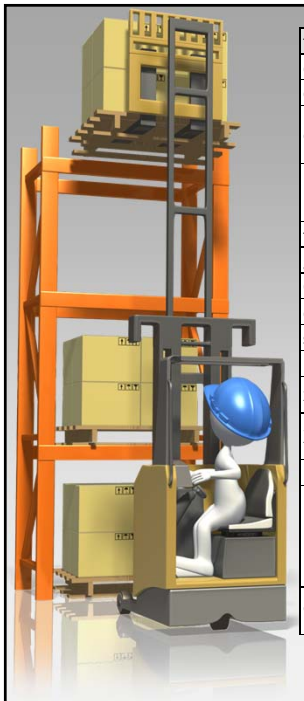
-: the result of the conversion is left-justified within the field

+: the sign of signed conversions is always prepended to the result

(space):If no sign is going to be written, a blank space is inserted before the value.

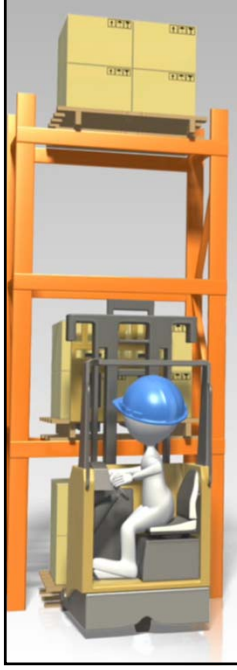
: *alternative form* of the conversion is performed_ "0x".

0 : for integer and floating point number conversions, leading zeros are used to pad the field instead of *space* characters.



type	Output	Example
c	Character	a
d or i	Signed decimal integer	392
e	Scientific notation (mantise/exponent) using e character	3.9265e+2
E	Scientific notation (mantise/exponent) using E character	3.9265E+2
f	Decimal floating point	392.65
g	Use the shorter of %e or %f	392.65
G	Use the shorter of %E or %f	392.65
o	Unsigned octal	610
s	String of characters	sample
u	Unsigned decimal integer	7235
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (capital letters)	7FA
p	Pointer address	B800:0000
n	Nothing printed. The argument must be a pointer to a signed int, where the number of characters written so far is stored.	
%	A % followed by another % character will write % to stdout.	%

{c,s} {l,d,o,u,x,X} {e,E,f,g,G}



printf() example

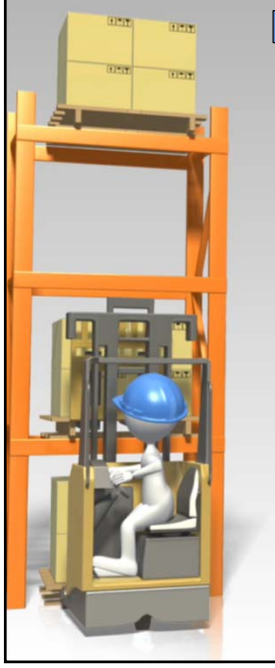
```
#include <stdio.h>
int main() {
printf ("Characters: %c %c \n", 'a', 65);
printf ("Decimals: %d %ld\n", 1977, 650000L);
printf ("Preceding with blanks: %10d \n", 1977);
printf ("Preceding with zeros: %010d \n", 1977);
printf ("Some different radixes: %d %x %o %#x %#o \n",
100, 100, 100, 100, 100);
printf ("floats: %4.2f %+.0e %E \n",
3.1416, 3.1416, 3.1416);
printf ("Width trick: %*d \n", 5, 10);
printf ("%s \n", "A string");
return 0; }
```

Characters: a A
Decimals: 1977 650000
Preceding with blanks: 1977
Preceding with zeros: 0000001977
Some different radixes: 100 64 144 0x64 0144
floats: 3.14 +3e+000 3.141600E+000
Width trick: 10
A string

fprintf example

```
#include <stdio.h>

int main () {
   FILE * pFile;
   int n;
   char name [100];
   pFile = fopen ("myfile.txt","w");
   for (n=0 ; n<3 ; n++)    {
     puts ("please, enter a name: ");
     gets (name);
     fprintf (pFile, "Name %d [%-10.10s]\n",n,name);
   }
   fclose (pFile);
   return 0; }
```

myfile.txt
Name 1 [John]
Name 2 [Jean-Franc]
Name 3 [Yoko]



Exercises #1

1.1. Code a program that input a odd whole numbers and output the character 8 which its high and wide are correspondent with the number inputted.

Ex: n = 5

```
*  *  *  *  *
*           *
*  *  *  *  *
*           *
*  *  *  *  *
```

1.2. Code a program that input a whole number n and output a triangle with its high is n

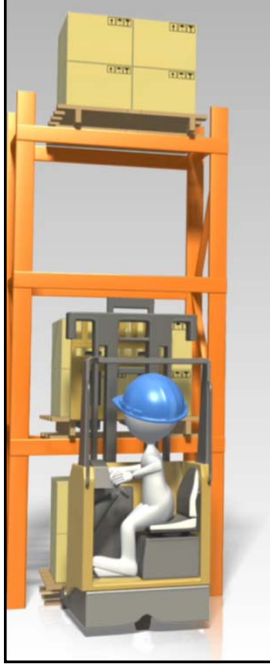
Ex:

n = 5

```
     *
    *  *
   *    *
  *      *
 * * * * * *
```

scanf(); fscanf(); sscanf();

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

int fscanf( FILE *stream, const char *format, ... );

int sscanf( const char *buffer, const char *format, ... );
```



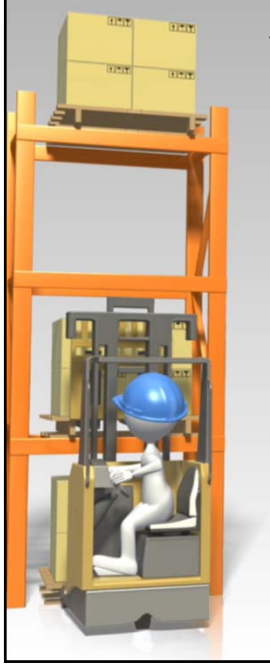
Format placeholders

Syntax:

%[*][*width*][modifiers]*type*

%[*][(number)] [h|l|L]type

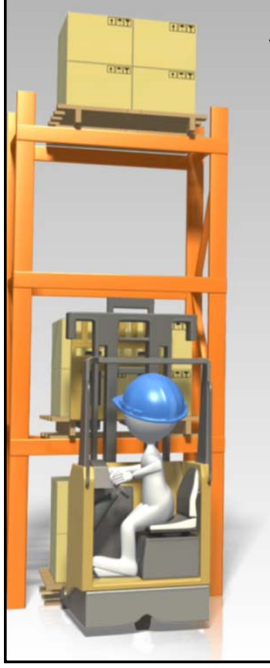
type	Qualifying Input	Type of argument
c	Single character:	char *
d	Decimal integer:	int *
e,E,f,g,G	Floating point:	float *
o	Octal integer.	int *
s	String of characters.	char *
u	Unsigned decimal integer.	unsigned int *
x,X	Hexadecimal integer.	int *



scanf() Example

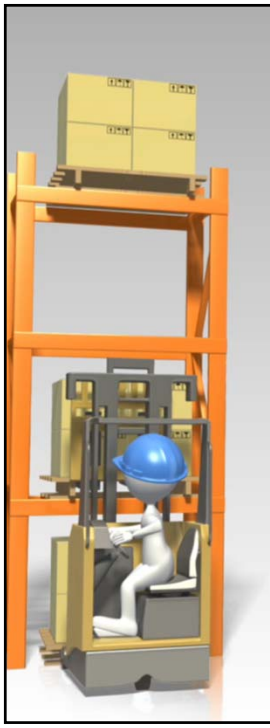
```
#include <stdio.h>
int main(int argc, char **argv) {
char input_array[100];
int i;
double d;
fprintf(stdout, "Enter your name: "); scanf("%s",input_array);
fprintf(stdout, "Enter a decimal value between 1 and 100: ");
 scanf("%d",&i);
fprintf(stdout, "Name: %s , %d is between 1 and  100.\n",input_array,i);
fprintf(stdout, "Enter a hexadecimal number: "); scanf("%x",&i);
fprintf(stdout, "You have entered %#x (%d).\n",i,i);
return 0; }
```

```
Enter your name: Erik
Enter a decimal value between 1 and 100: 45
Name: Erik , 45 is between 1 and 100.
Enter a hexadecimal number: beef
You have entered 0xbeef (48879).
```



Analyzing Strings with sscanf

```
float f;
int i,a[10];
char s1[25],s2[25];
char the_string[] = "foo -3.6 fum dum 17";
sscanf(the_string, "foo %f fum dum %d", &f, &i);

sscanf("foo 1 2 3 fum", "%s %d %d %s", s1, &a[0], &a[1],
&a[2], s2);

sscanf("foo", "f%s", s1);

sscanf("4711bar", "%d%2s", a, s1);

sscanf("We are the World!", "%s", s1);
sscanf(stdin, "%s", s1);
sscanf(stdin, "%[^\n]s", s1);

sscanf("12", "%D", i);
sscanf("014", "%D", i);
sscanf("0xC", "%D", i);
```



Exercises #2

2. Input a whole number n. Put the plus or minus signs ('+' or '-') between the numbers 1, 2, 3, 4, 5, 6, 7, 8, 9 have been written in order to create an expression which its value equals to n.

Ex: n=122 => 12 + 34 – 5 – 6 +78 + 9

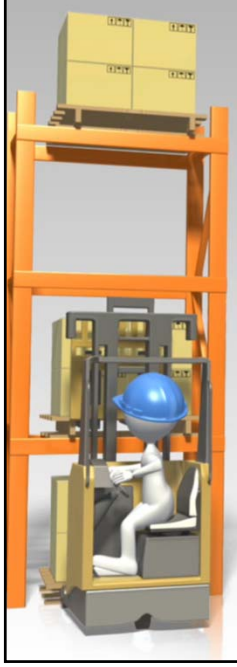
III. C's File I/O Streams

General framework for processing file in C language

fopen(); fclose();

fseek(); rewind();

fgets(); fputs(); fgetc(); fputc();



General Framework

1. DECLARE a FILE * variable

```
FILE *infile;
FILE *outfile;
```

2. OPEN the file

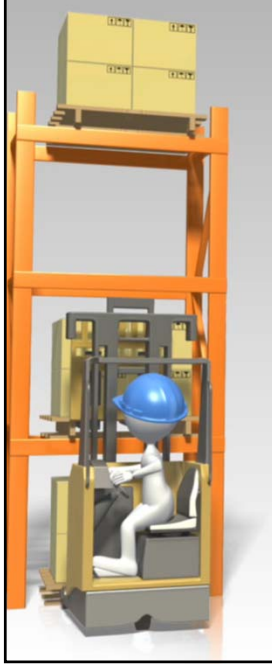
```
 // using relative path name of file
infile = fopen("input.txt", "r");
if (infile == NULL) { Error("Unable to open file."); }
// using absolute path name of file
outfile = fopen("/home/newhall/output.txt", "w");
if (outfile == NULL) {  Error("Unable to open file."); }
```

3. USE I/O operations to read, write, or move the current position in the file

```
int ch;
ch = getc(infile);
putc(ch, outfile);
```

4. CLOSE the file: use fclose() to close the file after you are done with it

```
fclose(infile);
fclose(outfile);
```



fopen(); fclose();

FILE *fopen(const char *filename, const char *mode);

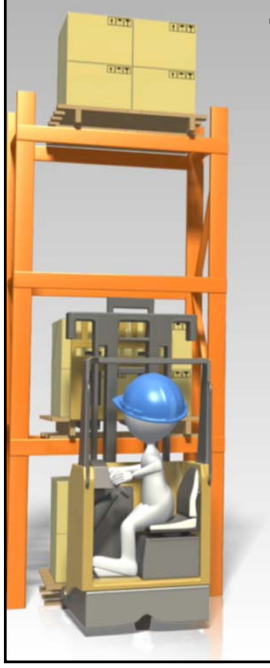
int fclose(FILE *stream);

Example:

```
#include <stdio.h>
int main () {
        FILE * pFile;
        pFile = fopen ("myfile.txt","wt");
        fprintf (pFile, "fclose example");
        fclose (pFile);
        return 0;
}
```

Mode parameter

File access mode string	Meaning	Explanation	Action if file already exists	Action if file does not exist
"r"	read	Open a file for reading	read from start	failure to open
"w"	write	Create a file for writing	destroy contents	create new
"a"	append	Append to a file	write to end	create new
"r+"	read extended	Open a file for read/write	read from start	error
"w+"	write extended	Create a file for read/write	destroy contents	create new
"a+"	append extended	Open a file for read/write	write to end	create new
{b|t}				



fseek(); rewind();

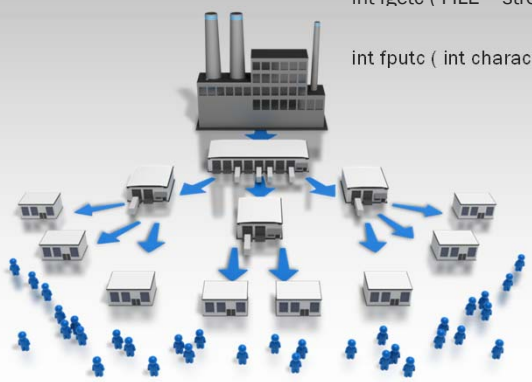
```
int fseek( FILE *stream, long offset, int origin );

void rewind( FILE *stream );
```

Example:

```
rewind(infile);
fseek(f, 0, SEEK_SET); // seek to the beginning of the file
fseek(f, 3, SEEK_CUR); // seek to 3 chars from the current position
fseek(f, -3, SEEK_END); // seek to 3 chars from the end of the file
```

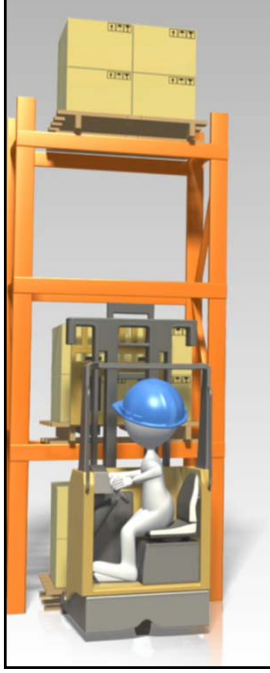
fgets(); fputs(); fgetc(); fputc();

```
char * fgets ( char * str, int num, FILE * stream );

int fputs ( const char * str, FILE * stream );

int fgetc ( FILE * stream );

int fputc ( int character, FILE * stream );
```



Example

```
#include <stdio.h>
int main() {
        FILE * pFile;
        char mystring [100];
        pFile = fopen ("myfile.txt" , "r");
        if (pFile == NULL)
                perror ("Error opening file");
        else {
                if ( fgets (mystring , 100 , pFile) != NULL )
                        puts (mystring);
                fclose (pFile);
        }
        return 0;
}
```



Exercises #3

3. Code a simple interpreter to control the movement of a robot. Input commands from a file .TXT and draw the path of the robot on the screen of your computer.

The commands are

Move to (*x*,*y*)

Turn left

Turn right

Go forward *n* steps

Go back *n* steps

Questions?

I'm happy to help you!

hvusynh@hcmiu.edu.vn