Data Structures

C I/O

Single Character Input: getchar()

```
#include <stdio.h>
main()
{
  int ch;
  ch = getchar() ;
}
```

Single Character Output: putchar()

```
#include <stdio.h>
main()
{  int ch;
   putchar('?') ;      /* display a prompt */
   ch = getchar() ;    /* read a keyboard char */
   putchar(ch) ;       /* echo to screen */
}
```


Printing One Line: prinf()

```
printf("Type any keyboard character.\n") ;
printf("\n Your character is %c \n",ch) ;
```

Special C Character Constants			
constant	ASCII name	hex value	meaning
\n	LF	0x0a	newline
\t	HT	0x09	tab
\v	VT	0x0b	vet tab
\b	BS	0x08	backspace
\r	CR	0x0d	return
\f	FF	0x0c	form feed
\\		0x5c	backslash
\"		0x22	quotation marks
\'		0x27	apostrophe
\0	NUL	0x00	null

Conversion Strings and Specifiers

printf("conversion string", variable list)

printf() Conversion Specifiers	
Specifier	meaning
%c	print a character
%d	print an integer
%i	print an integer (same as %d)
%e	print float value in exponential form
%f	print float value
%g	print using %e or %f, whichever is smaller
%o	print octal value (unsigned)
%s	print a string
%x	print hexadecimal integer (unsigned) using lowercase a-f
%X	print hexadecimal integer (unsigned) using uppercase A-F
%p	print a pointer value

The modifiers in the form

% - n.p l

-	left justify the argument
n	field width (optional)
.	separates field width from precision
p	number of places after decimal (precision)
l	indicates long type

Example:

```
n = 256
x = 129.546
ch = 'R'
printing  n   %3d        prints as    256
printing  n   %d         prints as     256
printing  n   %10d       prints as    256
printing  n   %f         prints as     256.000
printing  n   %-8d       prints as    256 with 5 trailing blanks
printing  n   %o         prints as     400
printing  n   %x         prints as     100
printing  n   %u         prints as     256
printing ch   %c         prints as     R
printing ch   %d         prints as     82
printing  x   %e         prints as     1.29546 E2
printing  x   %7.2f      prints as     129.55
printing  x   %.3f       prints as     129.546
```


Formatted Input: scanf()

scanf(" conversion string", variable list) ;

Scanf() Conversion Specifiers	
Specifier	Meaning
%c	reads a character
%d	reads an integer
%e	reads a float value
%f	reads a float value
%h	reads a short integer
%o	reads an octal value (unsigned)
%s	reads a string
%x	reads a hexadecimal integer (unsigned)
%u	reads an unsigned integer
%i	reads a decimal, octal or hexadecimal integer
%p	reads the hexadecimal representation of a pointer

```
scanf("%d", &n) ;
 scanf("%7.f,%3d",&average,&count) ;
 char name[20] ;
 scanf("%20s",name) ;
```

Reading a Line of Data: gets()

This is a sample string input.<return>

```
char inbuff[81] ;                    /* assume 80 col screen */
printf("What is the input?\n") ;
gets(inbuff) ;                                    /* read a line */
printf("You typed %s\n",inbuff) ;        /* echo */
```

The **inbuff** string looks like this:

This is a sample string input\0

The Return key (newline or \n) is not part of the string. The function **gets()** appends the null character.

The **gets()** function returns a value. *The value is a pointer.*

```
char *result, inbuff[81] ;          /* result is a pointer to */
result = gets(inbuff) ;             /* a memory address  */
printf("%s\n and result is %s\n",inbuff,result) ;
```

This is a sample string input.

and result is This is a sample string input.

If an error occurs in the input the gets() returns 0 (or NULL).

7

Printing a String: puts()

```
char  name[] ;
gets(name) ;
puts("This is a sample string.") ;
puts(name) ;
puts(&name) ;
```

The puts() function always executes a newline character when it is used. A special version, cputs() *does not write a newline when executed.*

```
result = getch() ;
ungetch(result) ;
```

8

```
#include <stdio.h>    /*  L3.C      example 1 */
main()
{  /* pause to input a symbol from keyboard */
  int r ;                                                    /* result */
  printf("To proceed - press a key + <CR>\n") ;
  r = getchar() ;                          /* entering a symbol */
  printf("The program continues!\n") ;
  printf("The received value from getchar is: %2d \n", r ) ;
}
```

```
#include <stdio.h>                          /* example 2 */
main()
{ int ch ;
  putchar('?') ;                            /* display a prompt */
  ch = getchar() ;   /* read a keyboard char " Hello <CR>" */
  putchar(ch) ;                             /* echo to screen   "H" */
}
```

9

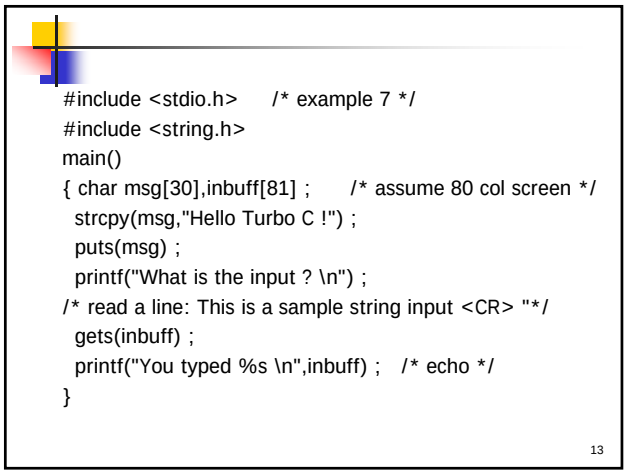
```
#include <stdio.h>                          /* example 10 */
main()
{ char name[30] ;
  gets(name) ;                                        /* Enter your first
    name */
  puts("This is a sample string") ;
  puts(name) ;
  puts(&name) ;
}
```

```
#include <stdio.h>   /*  example 6 */
 main()
 {int n ;
   char first_nm[20], last_nm[20] ;
   float cost ;
/*1*/printf("Please type in your first and last names.");
/*2*/scanf("%s %s",first_nm,last_nm) ;
/*3*/putchar('\n') ;
/*4*/printf("How many items are there?") ;
/*5*/scanf("%d",&n) ;
/*6*/putchar('\n') ;
/*7*/printf("What is the unit cost?") ;
/*8*/scanf("%f",&cost) ;
/*9*/printf("\n\nOk %s %s, \n There are %d items at $%4.2f\n",
    first_nm,last_nm,n,cost) ;
 }
```

11

```
#include <stdio.h>   /*  example 6 */
 main()
 {
   int n ;
   char first_nm[20], last_nm[20] ;
   float cost ;
/*1*/printf("Please type in your first and last names.");
/*2*/scanf("%s %s",first_nm,last_nm) ;
/*3*/putchar('\n') ;
/*4*/printf("How many items are there?") ;
/*5*/scanf("%d",&n) ;
/*6*/putchar('\n') ;
/*7*/printf("What is the unit cost?") ;
/*8*/scanf("%f",&cost) ;
/*9*/printf("\n\nOk %s %s, \n There are %d items at $%4.2f\n",
    first_nm,last_nm,n,cost) ;
 }
```

12



```
#include <stdio.h>      /* example 7 */
#include <string.h>
main()
{ char msg[30],inbuff[81] ;      /* assume 80 col screen */
  strcpy(msg,"Hello Turbo C !") ;
  puts(msg) ;
  printf("What is the input ? \n") ;
/* read a line: This is a sample string input <CR> "*/
  gets(inbuff) ;
  printf("You typed %s \n",inbuff) ;   /* echo */
}
```

13