Character Data Type

char Data Type

- **char** keyword is used to refer character data type.
- Character data type allows a variable to store only one character.
- Syntax:

variable declaration
char ch;

variable initialization
ch = ‘a’;

variable declaration + initialization
char ch = ‘a’;

Example

```
#include<stdio.h>
void main() {
        char flag = 'a' ;
        printf("%c \n",flag);
}
```

Output:

a

```
#include<stdio.h>
void main() {
        char flag = ‘n’ ;
        printf(“Enter a character:\n”);
        scanf(“%c”, &flag);
        printf(“Entered character is:%c
\n",flag);
}
```

Output:

a

ASCII and Why?

- ASCII stands for **American Standard Code for Information Interchange**
- A computer system normally stores characters using the ASCII code.
- Each character is stored using eight bits of information, giving a total number of 256 different characters (2**8 = 256).

List of ASCII characters

0X....
0x.....
0o....
0O....

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Example

```
#include <stdio.h>
void main() {
        char c;
        printf("Enter a character: ");
        scanf("%c", &c);

        // %d displays the integer value of a character
        // %c displays the actual character
        printf("ASCII value of %c = %d", c, c);

}
```

ctype.h functions

- isalpha(ch)
- isdigit(ch)
- isalnum(ch)
- ispunct(ch) ‘ “ ‘!’ ?
- isspace(ch)
- isupper(ch) A upper a lower
- islower(ch) ‘a’ -> true 97 to 97+26 ‘A’ -> false ‘!’ -> false
- toupper(ch) toupper(‘A’) = A toupper(‘a’) = A
- tolower(ch) tolower(‘a’) = a tolower(‘A’) = a
- “Enter y to continue!” tolower(‘Y’) = y var ==‘y’

Example

```
#include<ctype.h>
#include<stdio.h>
void main()
{
        char n;
        printf("\nEnter a character=");
        n=getche();

        if(isalpha(n))
                printf("\nYou typed an alphabet");
        if(isdigit(n))
                printf("\nYou typed a digit");
        if(isalnum(n))
                printf("\nYou typed an alphabet or digit");
        if(isspace(n))
                printf("\nYou typed a blank space");
        if(ispunct(n))
                printf("\nYou typed punctuator");
}
```

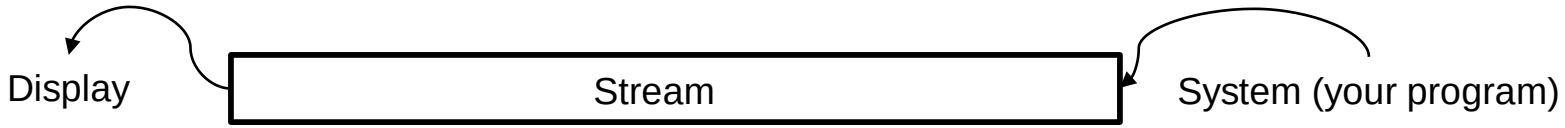
I/O Streams stdin and stdout

What is stream?

- A stream is a sequence of characters with functions to take characters out of one end, and put characters into the other end.
- In the case of input/output streams, one end of the stream is connected to a physical I/O device such as a keyboard or display.

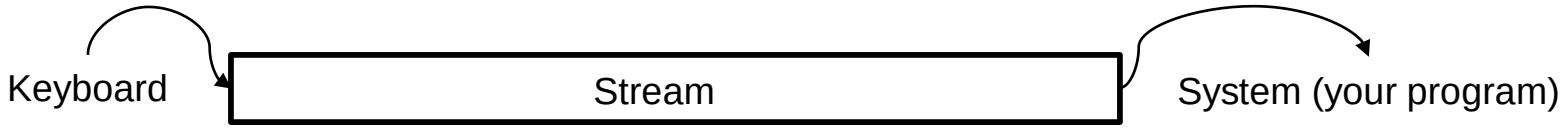
Output stream (stdout)

- If it is a console output stream, your program puts characters into one end of the stream, and the display system takes characters out of the other and puts them on the screen.



Input stream (stdin)

- If it is a console input stream, the keyboard puts characters into one end of the stream, and your program takes characters out of the other and stores the results in variables in the program.



Contd.

- If no characters are waiting in the input stream, your program must wait until you supply some by typing on the keyboard.
- File streams follow the same principle, except that the file system is attached to the other end of the stream.

Problem

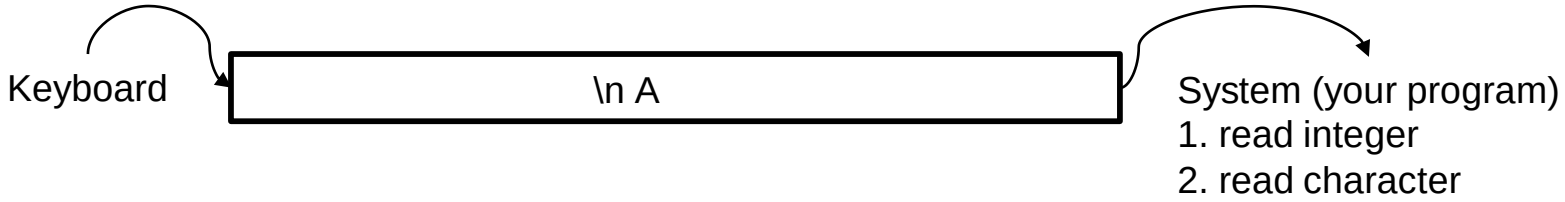
```
#include<stdio.h>
void main() {
        int num;
        char ch;
        printf(“\nEnter a number:”);
        scanf(“%d”, &num);
        printf(“\nEntered number is:%d”, num);

        printf(“\nEnter a character:”);
        scanf(“%c”, &ch);
        printf(“\nEntered character is:%c”, ch);
}
```

Output:
Enter a number:12<Enter>

Entered number is: 12
Enter a character:
Entered character is:

Why?


Keyboard
\n A
System (your program)
1. read integer
2. read character

Solution

- Using scanf’s formatted feature

```
scanf(“%d”, &num);
scanf(“ %c”, &ch);
```

- Changing the sequence of input

```
scanf(“%d”, &num);
scanf(“%c”, &ch);

scanf(“%c”, &ch);
scanf(“%d”, &num);
```

- In almost all cases, the scanf function starts the scan for the value by skipping over any initial whitespace characters.

Contd

- Using the following cleaning code

while ((ch = getchar()) != '\n’ && ch != EOF); //EOF is constant with value -1

Overflow and a way out

Overflow in Variables

- The data types like int, char, long can hold bounded values.
- A complex expression that produces a final value within bound might produce intermediate values that go beyond the bounds
  - Overflow
  - May result in incorrect final value
- Some tricks or simplification may be needed to get correct value.

Avoiding Overflow: Examples

- Permutation, r out of n: n! / (n-r)!
- Computation of 100! / 98!
  - 100! And 98! are too big for computer
  - But the result 100*99 can be computed easily

- To avoid overflow: write smart algorithm

Avoiding Overflow: Examples

- Terms in the series: T(n) = (x + 1)^2*n / (2*n + 1)!
- Direct computation of nth term:
  - May not fit in the data type
- Better way to compute:
  - T(n) = T(n-1) * R
  - Where R = (x+1)^2 / 2*n * (2*n + 1)
- Can calculate this for large n

Program

```
#include<stdio.h>
void main() {
        float x, term = 1.0;
        int i;
        scanf(“%f”, &x);

        for( i=1; i<=50; i++ ) {
                term = term * ((x + 1) * (x + 1) / (2 * i) * (2 * i + 1));
                printf(“The %dth term is: %f”, i, term);
        }
}
```

Variable Scope

Scope of variable in C

```
#include<stdio.h>
void main() {

        for(int i=1; i<=2; i++)
                i += 1;
                printf(“%d\n”, i);
}
```

What will be the output?

Scope of variable in C

```
#include<stdio.h>
void main() {

        for(int i=1; i<=2; i++)
                i += 1;
                printf(“%d\n”, i);
}
```

What will be the output?

Compiler Error

Same program using while loop

```
#include<stdio.h>
void main() {
        int i;
        while(i<=2) {
                i +-= 1;
        }
         printf(“%d\n”, i);
}
```

What will be the output now?
Again compiler error?
No. This time it will compile successfully.

Block scope of a variable

```
#include<stdio.h>
void main() {
        {       \\start block
                int i;
                for ( i=1; i<=2; i++ )
                        printf("%d\n", i);
        }       \\end block
}
```

Output?
1
2

Contd.

```
#include<stdio.h>
void main() {
        {           \\start block
                    int i;
                    for ( i=1; i<=2; i++ )
                                printf("%d\n", i);
        }           \\end block
         printf("outside %d\n", i);
}
```

Output?
Compiler Error: i undeclared

Another example

```
#include<stdio.h>
void main() {
        int i;
        for ( i=1; i<=2; i++) {
                int j;
                if (i==1) {
                        j = 0;
                }
                j = j + 1;
                printf("i=%d, j=%d", i, j);
        }
}
```

Output:
i=1, j=1
i=2, j=???