"C" PROGRAMING

owner

GURU MUKH

for BCA 1st year

विषय सूची

विषय सूची

विषय सूची

विषय सूची

विषय सूची

विषय सूची

विषय सूची

Introduction of C

C Programming Introduction

- C Programming ये language सीखने में बहुत ही आसान Language है |
- C Language में बहुत बड़े-बड़े software बनाए गए है जैसेकि, Operating Systems, Databases, Computer Drivers, Editors, Assembler, Compiler etc.
- C Programming के बनाने का उद्देश Unix Operating System बनाना ही था, लेकिन आज ये language सबसे ज्यादा इस्तेमाल होनेवाली language है |
- जो कोई C Programming अच्छी तरह से सीखता है तो उसे ज्यादातर कोई दूसरी language सिखने में कोई दिक्कत नहीं आती जैसीकि, C++, Core Java या Python.
- Computer के या अन्य किसी Hardware को चलाने के लिए C Language का ज्यादातर इस्तेमाल होता है |
- Computer के या अन्य किसी Hardware को चलाने के लिए C Language का ज्यादातर इस्तेमाल होता है |
- C Programming ये एक General-Purpose High-Level Language है |

C Programming History

- C Programming के अविष्कार से पहले Martin Richards ने 1966 में BCPL मतलब Basic Combined Programming Language का जनम हुआ |
- बाद में Ken Thompson ने BCPL को और develop करके BCPL के पहले अक्षर का मतलब B Language का अविष्कार किया |
- B Language के चलते Ken Thompson ने Basic तरह की Unix Operating System बनाई |
- उसके बाद Dennis Ritchie ने B Language को और develop करके Bell Laboratory में C Programming Language बनाई |
- C Programming को बनाने के बाद Dennis Ritchie और Ken Thompson ने मिलकर UNIX Operating System को develop किया |

C Programming Reserved Keywords

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

Data Types	**int :** Integer type के variable को declare करने के लिए 'int' data type का इस्तेमाल किया जाता है | for e.g. int a=0; 'a' is integer variable. **char :** Character type के variable को declare करने के लिए 'char' data type का इस्तेमाल किया जाता है | for e.g. char ch='Hello'; 'ch' is character variable. **double :** Double type के variable को declare करने के लिए 'double' data type का इस्तेमाल किया जाता है | इसका उपयोग ज्यादातर नहीं होता , इसके बदले में float data type का इस्तेमाल होता है | for e.g. double num=3.5; 'num' is double variable. **float :** Floating type के variable को declare करने के लिए 'float' data type का इस्तेमाल किया जाता है | for e.g. float var=1.20; 'var' is floating variable.

Loop	Example	Output

do...**while** Loop	void main(){
	int i=0;
	do{
	printf("%d\n",i);
	i++;
	}
	while(i<5);
	}
		0
		1
		2
		3
		4

for Loop	void main(){
	int i;
	for(i=0; i<5; i++){
	printf("%d\n",i);
	}
	}
		0
		1
		2
		3
		4

Decision Control Statements	Example	Output

if...else	void main(){
	int i=10;
	if(i==10){
	printf("i is equal to 10");
	}else{
	printf("i is not equal to 10");
	}
	}
		i is equal to 10

switch.. case.. default	void main(){
	
	switch(2){
	
	case 1:
	printf("expression is 1");
	break;
	case 2:
	printf("expression is 2");
	break;
	default:
	printf("unknown expression");
	}
	}
		expression is 2

Storage Class	Example	Output

auto	void main()
	{
	auto int num = 10 ;
	{
	auto int num = 5 ;
	printf("num : %d\n",num);
	}
	printf("num : %d",num);
	}
		num : 5
		num : 10

extern	extern int num;
	void main()
	{
	printf("num value comes from General File is %d", num);
	}
		num value comes from General File is 10

register	void main()
	{
	register int num=1;
	printf("These Number is stored in CPU's register\n");
	printf("%d ",num);
	}
		These Number is stored in CPU's register
		1

static	function(){
	static int num = 1;
	printf("%d",num);
	num++;
	}
	void main(){
	int i;
	for(i=0; i<10; i++){
	printf("\n");
	function();
	}
	}
		1
		2
		3
		4
		5
		6
		7
		8
		9
		10

Jumping Statements	Example	Output

goto	void main(){
	int num1, num2, num3;
	char ch;
	yes : printf("Enter two values\n");
	scanf("%d%d",&num1, &num2);
	num3 = num1 + num2 ;
	printf("Addition of %d and %d is %d\n", num1, num2, num3);
	printf("\nDo you want to continue y(yes) or n(No)");
	scanf(" %c",&ch);
	if(ch=='y'){
	goto yes;
	}
	else if(ch=='n'){
	goto no;
	}
	else{
	printf("You entered other key");
	}
	no : printf("Do you want to exit ?, Press Enter");
	}
		Enter two values
		5
		6
		Addition of 5 and 6 is 11
		
		Do you want to continue y(yes) or n(No)n
		Do you want to exit ?, Press Enter

continue	void main(){
	int i=0;
	while (i < 20){
	printf("value of i is %d\n", i);
	i++;
	if (i == 10)
	printf("Continued Values\n");
	continue;
	}
	}
		value of i is 0
		value of i is 1
		value of i is 2
		value of i is 3
		value of i is 4
		value of i is 5
		value of i is 6
		value of i is 7
		value of i is 8
		value of i is 9
		Continued Values
		value of i is 10
		value of i is 11
		value of i is 12
		value of i is 13
		value of i is 14
		value of i is 15
		value of i is 16
		value of i is 17
		value of i is 18
		value of i is 19

struct	struct Student {
	char stud_name[30];
	};
	void main(){
	struct Student info;
	strcpy(info.stud_name, "Pradeep Rana");
	printf("Student name is %s ", info.stud_name);
	}
		Student name is Pradeep Rana

return	Number(){
	int a=10, b=5, c;
	c = a + b;
	return c;
	}
	void main(){
	printf("Addition of a and b is %d",Number());
	}
		Addition of a and b is 15

typedef	void main(){
	typedef int num;
	num num1;
	printf("Enter value of num1 ");
	scanf("%d", &num1);
	printf("Value of num1 is %d", num1);
	}
		Enter value of num1 5
		Value of num1 is 5

union	union Student {
	char stud_name[30];
	};
	void main(){
	union Student info;
	strcpy(info.stud_name, "Pradeep Rana");
	printf("Student name is %s ", info.stud_name);
	}
		Student name is Pradeep Rana

const	void main(){
	const int num=1;
	printf("num = %d",num);
	}
		num = 1

enum	enum Days {Sun, Mon, Tue, Wed, Thu, Fri, Sat };
	void main()
	{
	printf("%d, %d, %d, %d, %d, %d, %d",Sun, Mon, Tue, Wed, Thu, Fri, Sat);
	}
		0, 1, 2, 3, 4, 5, 6

sizeof	void main(){
	printf("%d",sizeof(int));
	}
		4

Structure of Program

preprocessor	#include <stdio.h>
return_type & main function	int main()
Opening curly brace	{
printf i/o function	printf("Hello World");
return 0;	return 0;
Closing curly brace	}

ये Program C Programming में सबसे पहला और आसान Program है | ऊपर दिया हुआ Program छह भाग में बटा हुआ है |

Preprocessor
main function
{
printf("Hello World");
return 0;
}

1.**Preprocessor** : Program में सबसे पहले preprocessors/header को लिखा जाता है | ये preprocessors अलग-अलग काम के लिए विभाजित किये हुए है,
for eg. stdio.h में printf और scanf function आते है |
conio.h में getch function आता है |
और भी कुछ preprocessors है |

2.**int main()** : यहाँ पर main function का return_type integer है | main function का return_type void भी लिखा जाता है पर void कोई भी value return नहीं करता | Program की शुरुआत main() function से होती है |

3.**{** : हर function के codes या statements को curly brace open होने के बाद लिखा जाता है |

4.**printf("Hello World");** : printf() function में लिखे हुए statement को output में print किया जाता है | इस statement को दो Double Quotes (" ") के अंदर लिखा जाता है |

5.**return 0;** : return 0 ये Program को बंद करने की अनुमति देता है | ये '0' main function को return करता है |

6.} : यहाँ पर } इस curly brace से main function को close किया है |

C Hello Program

Source Code :

```
#include <stdio.h>

int main(){

printf("Hello World !");

return 0;
}
```

Output :

```
Hello World !
```

How to Run C Program?

- Turbo C / C++ Download करे |
- Download किये हुए Turbo C / C++ को अपने Computer में install करे |
- उस Application को Open करके C Hello का Program लिखे |
- Program लिखने के बाद File पर जाकर save या F2 दबाया तो एक dialog box खुल जाएगा |
- Dialog Box खुलने के बाद उसे कोई भी नाम देकर उसे .c का extension दे | for eg. hello.c
- Save करने के बाद उसे F9 से Compile करे |
- अगर Program बिना error का हो तो CTRL+F9 से करे |

Data Types

C Integer- Basic Data Type

- Integer Data Type में variable को declare करने के 'int' keyword का इस्तेमाल करते है |
- Integer Data Type सभी numeric values को store कर सकता है |
- integer को output में print करना हो तो '%d' format specifier से output में print करते है |
- Integer Data Type 2, 4 और 8 bytes के हो सकते है |
- अगर Computer का Processor 16-bit हो तो int का size 2 Bytes होता है |
- अगर Computer का Processor 32-bit हो तो int का size 4 Bytes होता है |
- अगर Computer का Processor 64-bit हो तो int का size 8 Bytes होता है |

Date Type	Storage Size	Range
int (16-bit)	2 Bytes	-32,768 to 32,767
int (32-bit)	4 Bytes	-2,147,483,648 to 2,147,483,647
int (64-bit)	8 Bytes	-9,223,372,036,854,775,807 to 9,223,372,036,854,775,807
unsigned int (16-bit)	2 Bytes	0 to 65,535
short int	2 Bytes	-32,768 to 32,767
unsigned short int	2 Bytes	0 to 65,535
signed short int	2 Bytes	-32,768 to 32,767
long int	4 Bytes	−2,147,483,647 to 2,147,483,647
unsigned long int	4 Bytes	0 to 4,294,967,295
signed long int	4 Bytes	−2,147,483,648 to 2,147,483,647

sizeof इस keyword से int Data Types के size का पता करे | निचे दिया हुआ source code देखे |

Source Code :

```
#include <stdio.h>
int main() {
printf("int(16-bit) storage size : %d \n", sizeof(int));
printf("int(32-bit) storage size : %d \n", sizeof(int));
printf("int(64-bit) storage size : %d \n", sizeof(int));
printf("unsigned int storage size : %d \n", sizeof(unsigned int));
printf("short int storage size : %d \n", sizeof(short int));
printf("unsigned short int storage size : %d \n", sizeof(unsigned short int));
printf("signed short int storage size : %d \n", sizeof(signed short int));
printf("long int storage size : %d \n", sizeof(long int));
printf("unsigned long int storage size : %d \n", sizeof(unsigned long int));
printf("signed long int storage size : %d \n", sizeof(signed long int));
return 0;
}
```

Output :

```
int(16-bit) storage size : 2
int(32-bit) storage size : 4
int(64-bit) storage size : 8
unsigned int storage size : 4
short int storage size : 2
unsigned short int storage size : 2
signed short int storage size : 2
long int storage size : 4
unsigned long int storage size : 4
signed long int storage size : 4
```

C Character- Basic Data Type

- Character Data Type में variable को declare करने के 'char' keyword का इस्तेमाल करते है |
- Character को output में print करना हो तो '%c' format specifier से output में print करते है |
- सिर्फ एक ही character को declare कर सकते है | for eg. 'H'
- अगर multiple character मतलब पूरे एक string को print करना हो तो '%s' का इस्तेमाल करते है |
- Character Data Type 1 byte का होता है | |

Date Type	**Storage Size**	**Range**
char	1 Byte	-128 to 127
unsigned char (16-bit)	1 Byte	0 to 255
signed char	1 Byte	-128 to 127

Source Code :

```
#include <stdio.h>
 int main() {
 char str1='H'; // declare varible in single quotes
 char str2="H"; // declare varible in double quotes
 char str3[10]='Hello'; // Get Warning Error
 char str4[10]="Hello";
 printf("Single quoted Print Character: %c \n", str1);
 printf("Double quoted Print Character: %c \n", str2);
 printf("Single quoted Print String : %s \n", str3); //Get Warning Error
 printf("Double quoted Print String : %s \n", str4);
 return 0;
 }
```

Note : ऊपरवाला Program Warning Error देगा |Program में // Get Warning Error की lines हटाकर नीचेवाला Output दिया है | User single quotes और double quotes के बिच का फर्क जानने के लिए ऊपरवाला Program दिया है |

Output :

```
Single quoted Print Character: H
Double quoted Print Character: $
Double quoted Print String : Hello
```

sizeof इस keyword से char Data Types के size का पता करे | निचे दिया हुआ source code देखे |

Source Code :

```
#include <stdio.h>
int main() {
printf("char storage size : %d \n", sizeof(char));
printf("unsigned char storage size : %d \n", sizeof(unsigned char));
printf("signed char storage size : %d \n", sizeof(signed char));
return 0;
}
```

Output :

```
char storage size : 1
unsigned char storage size : 1
signed char storage size : 1
```

C Float - Basic Data Type

- Floating-point Data Type में variable को declare करने के 'float' keyword का इस्तेमाल करते है |
- float को output में print करना हो तो '%f' format specifier से output में print करते है |
- Floating- point Data Type 4 bytes का होता है | |

Date Type	Storage Size	Range	Digits of Precision
float	4 Bytes	1.2E-38 to 3.4E+38	6

sizeof इस keyword से float Data Types के size का पता करे | निचे दिया हुआ source code देखे |

Source Code :

```
#include <stdio.h>
int main() {
printf("Floating-point Storage size : %d \n", sizeof(float));
return 0;
}
```

Output :

```
Floating-point Storage size : 4
```

C Double - Basic Data Type

- Double Data Type में variable को declare करने के 'double' keyword का इस्तेमाल करते है |
- integer को output में print करना हो तो '%f', '%e', '%E', '%lf' format specifiers से output में print करते है |
- Double Data Type 8 bytes का होता है |
- Double और Floating-point Data Type में कोई फर्क नहीं है |

Date Type	Storage Size	Range	Digits of Precision
double	8 Bytes	2.3E-308 to 1.7E+308	15
long double	10 Bytes	3.4E-4932 to 1.1E+4932	19

Source Code :

```
#include <stdio.h>
int main() {
double a=5;
printf("Value of Double variable : %lf \n", a);
printf("Value of Double variable : %e \n", a);
printf("Value of Double variable : %E \n", a);
return 0;
}
```

Output :

```
Value of Double variable : 5.000000
Value of Double variable : 5.000000e+000
Value of Double variable : 5.000000E+000
```

sizeof इस keyword से double Data Types के size का पता करे | निचे दिया हुआ source code देखे |

Source Code :

```
#include <stdio.h>
int main() {
printf("Double Storage size : %d \n", sizeof(double));
return 0;
}
```

Output :

```
Double Storage size : 8
```

C Void - Basic Data Type

- void मतलब null value |
- void में कोई value नहीं होती |
- ये data type function और function के parameter / argument के लिए इस्तेमाल करते है |
- ये कोई value return नहीं करता |

Source Code :

```
#include <stdio.h>
void hello(void); // function with no return value and parameter
main()
{
hello();
}
void hello(void)
{
printf("Hello World");
}
```

Output :

```
Hello World
```

C Array - Derived Data Type

Array के लिए यहाँ क्लिक करे |

C Pointer - Derived Data Type

Pointer के लिए यहाँ क्लिक करे |

C Structure - User-defined Data Type

Structure के लिए यहाँ क्लिक करे |

C Union - User-defined Data Type

Union के लिए यहाँ क्लिक करे |

Constants & Variables

C Constants

- Constant की value fixed होती है |
- Constant किसी भी data type का हो सकता है |
- Constant को Literals भी कहते है |
- Constant Pointer भी होता है |

Syntax :

```
const data_type variable_name = value (optional) ;
```

Types of Constant :

- Integer Constant
- Decimal Constant
- Octal Constant
- Hexadecimal Constant
- Floating-point / Real Constant
- Character Constant
- String Constant
- Preprocessor

Constant Types	**With Examples**
Integer	2, 10, -2
Decimal (Integer)	2, 10, -2
Octal (Integer)	02, 010
Hexadecimal (Integer)	0x12, 0x1f
Floating-point/Real	-2.4, 4.8
Character	'i', 'j'
String	"Hello", "Hi"
Preprocessor	#define a 5

Integer Constant Types :

- Decimal Integer Constant
- Octal Integer Constant
- Hexadecimal Integer Constant

Integer Constants

- Integer Constant normal Variable की तरह काम करता है |
- Integer Constant positive (+) या negative (-) हो सकता है |
- Integer Constant की Range -32768 से 32767 तक होती है |

Source Code :

```
#include <stdio.h>
int main(){
const int num = 5; // integer Constant
printf("integer constant value : %d", num);
return 0;
}
```

Output :

```
integer constant value : 5
```

Character Constants

- Character Constant normal Variable की तरह काम करता है |
- Character Constant सिर्फ single character लेता है |
- Escape Sequences के साथ भी character constant इस्तेमाल किया जाता है |

Escape Sequences	With Examples	Escape Sequences	Explaination
\'	Single Quotation Mark	\f	Form Feed
\"	Double Quotation Mark	\n	New line
\\	Backslash	\r	Carriage Return
\?	Question Mark	\h	Horizontal Tab
\a	Audible Bell	\v	Vertical Tab
\b	Backspace		

Source Code :

```
#include <stdio.h>
int main(){
const char ch = 'H'; // character constant
const char escape[]= "Hello\tWorld"; // Escape sequence - Horizontal Tab and string constant
printf("Character constant : %c\n", ch);
printf("String constant : %s\n", escape);
return 0;
}
```

Output :

```
Character constant : H
String constant : Hello	World
```

Floating-point Constant

- Floating-point Constant normal float variable की तरह ही काम करता है |
- Floating-point Constant में Decimal point (.) होता है |
- अगर Floatint-point Constant की value integer type की हो तो वो Decimal point (.) लेता है |

Source Code :

```
#include <stdio.h>
int main(){
const float num1 = 5; // floating-point constant with integer value
const float num2 = 3.525984; // Floating-point constant
printf("Floating-point constant : %f\n", num1);
printf("Floating-point constant : %f\n", num2);
return 0;
}
```

Output :

```
Floating-point constant : 5.000000
Floating-point constant : 3.525984
```

String Constant

- String Constant की value Double Quotation Mark (" ") के अंदर लिखी जाती है |
- String Constant Single और Multiple characters लेता है |
- String Constant Escape Sequences के साथ भी इस्तेमाल करते है |

Source Code :

```
#include <stdio.h>
int main(){
const char str1 []= "H"; // String Constant with single character
const char str2 []= "Hello World"; // Normal String constant
const char str3 [] = "Hello\nWorld"; // String Constant with Escape Sequence
printf("String Constant with single character : %s\n", str1);
printf("Normal String constant : %s\n", str2);
printf("String Constant with Escape Sequence : %s\n", str3);
return 0;
}
```

Output :

```
String Constant with single character : H
Normal String constant : Hello World
String Constant with Escape Sequence : Hello
World
```

Preprocessor Constant

- Preprocessor के साथ भी Constant value रख सकते है |

Source Code :

```
#include <stdio.h>
#define a 5 // Constant value with preprocessor
#define b 10 // Constant value with preprocessor
int main(){
printf("Value of a  : %d\n", a);
printf("Value of b  : %d", b);
return 0;
}
```

Output :

```
Value of a  : 5
Value of b  : 10
```

C What is Variable

- ये data types की values अपने अंदर store करके रखता है |
- Variable ये एक memory location का नाम भी होता है |

Rules for Variable

- Variable ये case-sensetive होता है | for eg int a और int A ये अलग-अलग variables है |
- Variable की शुरुआत किसी भी alphabet(a-z, A-Z) या underscore(_) से होती है |
- Variables का नाम alphanumeric हो सकता है | For eg. a1 = 5, var1, var2
- Variable ये space को allow नहीं करता |
- Variable name कोई भी C Keywords नहीं होता |

C Variable Declaration and initialization

Variable Declaration

- जब Variable declare होता है तब ये variable जिस data type का होता है, उसके हिसाब से Memory allocate करता है |
- Variable Declare होने के बाद ये अपने अंदर Garbage Value लेता है |
- **Garbage Value** : Garbage Value Variable को Compiler द्वारा दी जाती है |

Syntax for Single Variable Declaration

```
data_type single_variable_name;
```

For Example

```
int a;
```

Source Code :

```
#include <stdio.h>
int main(){
int a;
printf("Value of a : %d", a);
return 0;
}
```

Output :

```
Value of a : 27
```

Variable_name	a
Variable_value	27
Address	5454

Garbage Value

Syntax for Multiple Variable Declaration

```
data_type multiple_variable_name;
```

For Example

```
int a, b, c;
```

Source Code :

```
#include <stdio.h>
int main(){
int a, b, c;
printf("Value of a : %d\n", a);
printf("Value of b : %d\n", b);
printf("Value of c : %d", c);
return 0;
}
```

Output :

```
Value of a : 27
Value of b : 8
Value of c : 35
```

Variable_name	a	b	
Variable_value	27	8	Garbage Value
Address	5454	5458	4bytes/Variable

Variable Initialization

- जब Variable initialize होता है तब ये variable जिस data type होता है, उसके हिसाब से Memory allocate करता है |for eg. int for 2bytes(16-bit) | 4bytes(32-bit) | 8bytes(64-bit), char
- Variable intialization में Variable को normal value दी जाती है |
- Variable intialization में एक variable सिर्फ एक ही value लेता है for eg. int a = 5, 6 ; int a = 5;

Syntax for Single Variable Initialization

```
data_type single_variable_name = value;
```

For Example

```
int a=5;
```

Source Code :

```
#include <stdio.h>
int main(){
int a = 5;
printf("Value of a : %d", a);
return 0;
}
```

Output :

```
Value of a : 5
```

Variable_name	a
Variable_value	5
Address	5454

Syntax for Multiple Variable Initialization

```
data_type single_variable_name = value, single_variable_name = value;
```

For Example

```
int a=5, b=6;
```

Source Code :

```
#include <stdio.h>
int main(){
int a = 5, b = 6;
printf("Value of a : %d", a);
printf("Value of b : %d", b);
return 0;
}
```

Output :

```
Value of a : 5
Value of a : 6
```

Variable_name	a	b	
Variable_value	5	6	
Address	5454	5458	4bytes/Variable

C Variable Scopes

Variable Scope के दो प्रकार है |

- Local Variable
- Global Variable

Local Variable

- Local Variables function के अंदर होते है |
- Local Variables जिस function के अंदर होते है वह पर ही वो visible रहते है |
- Local Variables की default value 'garbage value' होती है |

Source Code :

```
#include <stdio.h>
int main(){
int a = 5, b = 6, c; // Local Variable
printf("Value of a : %d", a);
printf("Value of b : %d", b);
printf("Default Value of c : %d", c);
return 0;
}
```

Output :

```
Value of a : 5
Value of b : 6
Default Value of c : 6 // garbage value
```

Global Variable

- Global Variables function के बाहर होते है |
- Global Variables की visibility पूरे program में होती है |
- Global Variables की default value '0' होती है |

Source Code :

```
#include <stdio.h>
int a = 5, b = 6, c; // Global Variable
int main(){
printf("Value of a : %d", a);
printf("Value of b : %d", b);
printf("Default Value of c : %d", c);
return 0;
}
```

Output :

```
Value of a : 5
Value of b : 6
Default Value of c : 0
```

Operators

Arithmetic Operators

Operators	Explaination
+ (Addition)	ये दो Operands को add करता है ।
- (Subtraction)	ये right operand से left operand को निकाल देता है ।
* (Multiplication)	ये दो Operands को multiply करता है ।
/ (Division)	ये right operand द्वारा left operand को divide करता है ।
% (Modulus)	ये right operand द्वारा left operand को divide करके remainder निकालता है ।

Source Code :

```
#include <stdio.h>
int main() {
int a,b,c;
printf("Enter two numbers ");
scanf("%d%d",&a,&b);
c=a+b;
printf("Addition of a and b is %d\n",c);
c=a-b;
printf("Subtraction of a and b is %d\n",c);
c=a*b;
printf("Multiplication of a and b is %d\n",c);
c=a/b;
printf("Division of a and b is %d\n",c);
c=a%b;
printf("Remainder of a and b is %d\n",c);
return 0;
}
```

Output :

```
Enter two numbers 5
4
Addition of a and b is 9
Subtraction of a and b is 1
Multiplication of a and b is 20
Division of a and b is 1
Remainder of a and b is 1
```

Relational Operators

Operators	Explaination
< (less than)	एक Operand की value दूसरे Operand से कम हो तो ये true return करता है | for eg. num1=5; num2=6; num1 < num2
> (greater than)	एक Operand की value दूसरे Operand से ज्यादा हो तो ये true return करता है | for eg. num1=6; num2=5; num1 > num2
<= (less than or equal to)	एक Operand की value दूसरे Operand से कम हो या बराबर (equal) हो तो ये true return करता है | for eg. num1=5; num2=5; num1 <= num2
>= (greater than or equal to)	एक Operand की value दूसरे Operand से ज्यादा हो या बराबर (equal) हो तो ये true return करता है | for eg. num1=5; num2=5; num1 >= num2
== (equal to)	दो Operands जब बराबर(equal) होते है, तब ये true return करता है |
== (equal to)	दो Operands जब एक-दूसरे से अलग होते है, तब ये true return करता है |

Source Code :

```
#include <stdio.h>
int main(){
int a=6, b=5;
if(ab){
printf(" a is greater than b\n");
}else{
printf(" a is less than b\n");
}
if(a>=b){
printf(" a is greater than b\n");
}else{
printf(" a is less than b\n");
}
if(a==b){
printf(" a is equal to b\n");
}else{
printf(" a is not equal to b\n");
}
return 0;
}
```

Output :

```
a is greater than b
a is greater than b
a is greater than b
a is greater than b
a is not equal to b
```

Logical Operators

Operators	Explaination
&& (logical AND)	अगर दोनों conditions true हो तो ये true return करेगा | for eg. (5<6) && (6>5)
|| (logical OR)	अगर दोनों में से एक भी true है , तो ये true return करेगा | for eg. (5<6) || (6>5)
! (logical NOT)	अगर condition true हो तो ये उसे false कर देता है | for eg. !((5<6) && (6>5)) !((5<6) || (6>5))

Source Code :

```
#include <stdio.h>
int main(){
if((5<6) && (6>5)){
printf("Condition is true.\n");
}else{
printf("Condition is false.\n");
}
if((5<6) && (6>5)){
printf("Condition is true.\n");
}else{
printf("Condition is false.\n");
}
if(!((5<6) && (5>6))){
printf("Condition is true.\n");
}else{
printf("Condition is false.\n");
}
return 0;
}
```

Output :

```
Condition is true.
Condition is true.
Condition is true.
```

Bitwise Operators

Truth Table for &, |, ^

a	b	a & b	a | b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Operation on AND (a&b)

अगर a = 20, b = 12 हो तो,

Decimal Value	Binary Value
20	0 0 0 1 0 1 0 0
12	0 0 0 0 1 1 0 0
4	0 0 0 0 0 1 0 0

Operation on AND (a&b)

अगर a = 20, b = 12 हो तो,

Decimal Value	Binary Value
20	0 0 0 1 0 1 0 0
12	0 0 0 0 1 1 0 0
28	0 0 0 1 1 1 0 0

Operation on XOR(a^b)

अगर a = 20, b = 12 हो तो,

Decimal Value	Binary Value
20	0 0 0 1 0 1 0 0
12	0 0 0 0 1 1 0 0
24	0 0 0 1 1 0 0 0

Binary Left Shift(<<) and Right Shift(>>)

Left Shift(<<) for e.g. a=20; /* 0001 0100 */ a << 2 में numeric value के binary value में हर binary number को 2 binary numbers left side से shift करता है | for e.g.a=20; /* 0001 0100 */ तो इसका 0101 0000 मतलब 80 हो जायेगा |

Right Shift(>>) for e.g. a=20; /* 0001 0100 */ ये Left shift से बिलकुल उलट है | Right Shift a>> 2 में numeric value के binary value में हर binary number को 2 binary numbers right side से shift करता है | for e.g.a=20; /* 0001 0100 */ तो इसका 0000 0101 मतलब 5 हो जायेगा |

Complement Operator (~)

Operation on Complement(~)

Decimal Value	Binary Value
~20	0 0 0 0 1 1 0 0
243	1 1 1 1 0 0 1 1

यहाँ पर Output -13 आने के बजाय 243 आया ऐसा क्यों ?

2's Complement of 243 -(reverse of 243 in binary + 1)

Operation on 2's Complement (~)		
Decimal Value	Binary Value	2's Complement
243	1111 0011	-(0000 1100+1) = -(0000 1101) = -13(output)

Source Code :

```
#include <stdio.h>
int main() {
int a=20; /* 0001 0100 */
int b=12; /* 0000 1100 */
int c;
c=a&b;
printf("value of c is %d",c); /* 4 = 0000 0100 */
c=a|b;
printf("\nvalue of c is %d",c); /* 28 = 0001 1100 */
c=a^b;
printf("\nvalue of c is %d",c); /* 24 = 0001 1000 */
c=a<<2;
printf("\nvalue of c is %d",c); /* 80 = 0101 0000 */
c=a>>2;
printf("\nvalue of c is %d",c); /* 5 = 0000 0101 */
printf("\nvalue of b is %d",~b); /* -13 = 1111 0011 */
return 0;
}
```

Output :

```
value of c is 4
value of c is 28
value of c is 24
value of c is 80
value of c is 5
value of b is -13
```

Assignment Operators

Assignment Operators ग्यारह प्रकार के होते है |

Operators	Example
= (assignment)	c = a + b
+= (add assignment)	c += a same as c = c + a
-= (subtract assignment)	c -= a same as c = c - a
= (multiply assignment)	c *= a same as c = c * a
/= (divide assignment)	c /= a same as c = c / a
%= (modulus assignment)	c %= a same as c = c % a
&= (AND assignment)	c &= a same as c = c & a
|= (OR assignment)	c |= a same as c = c | a
^= (XOR assignment)	c ^= a same as c = c ^ a
<<= (Left Shift assignment)	c <<= a same as c = c << a
>>= (Right Shift assignment)	c >>= a same as c = c >> a

Source Code :

```
#include <stdio.h>
int main() {
int a=20,b=12;
b = a + b;
printf("value of b is %d\n",b);
b += a;
printf("value of b is %d\n",b);
b -= a;
printf("value of b is %d\n",b);
b *= a;
printf("value of b is %d\n",b);
b /= a;
printf("value of b is %d\n",b);
b %= a;
printf("value of b is %d\n",b);
b &= 2;
printf("value of b is %d\n",b);
b |= 2;
printf("value of b is %d\n",b);
b ^= 2;
printf("value of b is %d\n",b);
b <<= 2;
printf("value of b is %d\n",b);
b >>= 2;
printf("value of b is %d\n",b);
return 0;
}
```

Output :

```
value of b is 32
value of b is 52
value of b is 32
value of b is 640
value of b is 32
value of b is 12
value of b is 0
value of b is 2
value of b is 0
value of b is 0
value of b is 0
```

Increment and Decrement Operators

I**ncrement Operator (++)** ये variable की value 1 से बढ़ा देता है |

Decrement Operator (--) ये variable की value 1 से घटा देता है |

Operators	**Same as**
++a (Increment Prefix)	a = a + 1
--a (Decrement Prefix)	a = a - 1
a++ (Increment Postfix)	
a-- (Decrement Postfix)	

Source Code :

```
#include <stdio.h>
int main() {
int a=20;
printf("Print Value with prefix : %d\n", ++a); // increase value with increment prefix
printf("Value of a : %d\n", a);
printf("Print Value with prefix : %d\n", --a); // decrease value with decrement prefix
printf("Value of a : %d\n", a);
printf("Print Value with postfix : %d\n", a++); // increase value with increment postfix
printf("Value of a : %d\n", a);
printf("Print Value with postfix : %d\n", a--); // decrease value with decrement postfix
printf("Value of a : %d\n", a);
return 0;
}
```

Output :

```
Print Value with prefix : 21
Value of a : 21
Print Value with prefix : 20
Value of a : 20
Print Value with postfix : 20
Value of a : 21
Print Value with postfix : 21
Value of a : 20
```

Conditional Or Ternary Operators

Conditional Operator में तीन Expressions होते है |
Conditional Operator को Ternary Operator भी कहते है |
Conditional Operator में अगर पहला expression true होता है, तो वो दूसरा expression output में print करता है |
अगर Conditional Operator में पहला expression false होता है, तो वो तीसरा expression output में print करता है |

Syntax for Conditional / Ternary Operator

```
expression1 ? expression 2 : expression 3
```

Source Code :

```
#include <stdio.h>
int main()
{
int a = 100, b ;
b = ( a == 100 ? 2 : 0 ) ;
printf("Value of a is %d\n", a);
printf("Value of b is %d\n", b);
return 0;
}
```

Output :

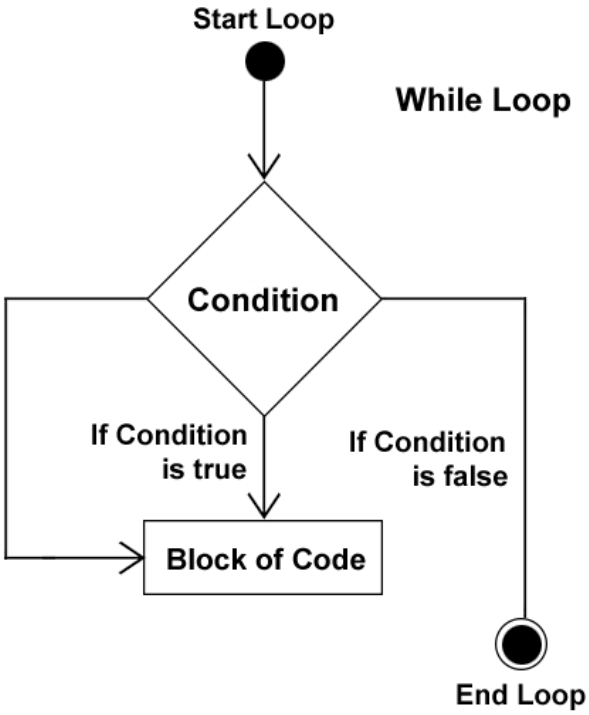
```
Value of a is 100
Value of b is 2
```

C While Loop

- While Loop में variable को initialize करना जरुरी है |
- अगर While Loop को repeat करना हो तो, increment/decrement operator का इस्तेमाल किया जाता है |
- जबतक While Loop में Condition true होती तब तक repeat होता रहता है |

Syntax :

```
variable initialization ;
while(condition){
statements;
variable increment/decrement;
}
```



For Example :

```
while(i<10)
 {
    printf("%d\n",i); // statement of while loop
    i++; //increment operator
 }
```

Source Code :

```
#include <stdio.h>
int main(){
  int i=0; // Variable initialization
  while(i<10) // condition of while loop
  {
     printf("%d\n",i); // statement of while loop
     i++; //increment operator
  }
return 0;
}
```

Output :

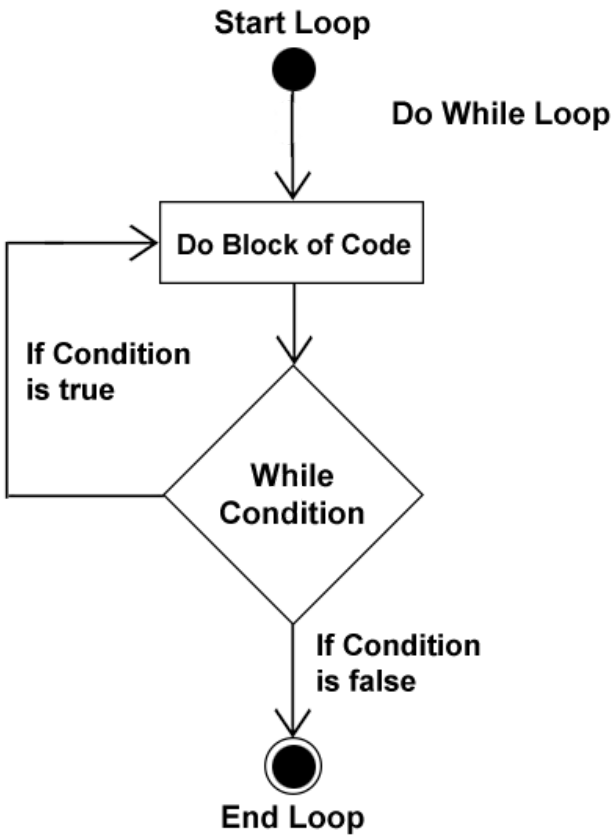
```
0
1
2
3
4
5
6
7
8
9
```

C Do While Loop

- जबतक Do-While Loop में Condition true होती तब तक repeat होता रहता है |
- Do-While की खासियत ये है कि, अगर condition false भी हो तो ये एक statement को output में print करता है |

Syntax :

```
do{
statements;
}while(condition);
```



For Example :

```
do
  {
     printf("%d\n",i);
     i++;
  }while(i<10);
}
```

Source Code :

```
#include <stdio.h>
int main()
{
  int i=0; // Variable initialization
  do
  {
     printf("%d\n",i);
     i++;
  }while(i<10);

return 0;
}
```

Output :

```
0
1
2
3
4
5
6
7
8
9
```

C For Loop

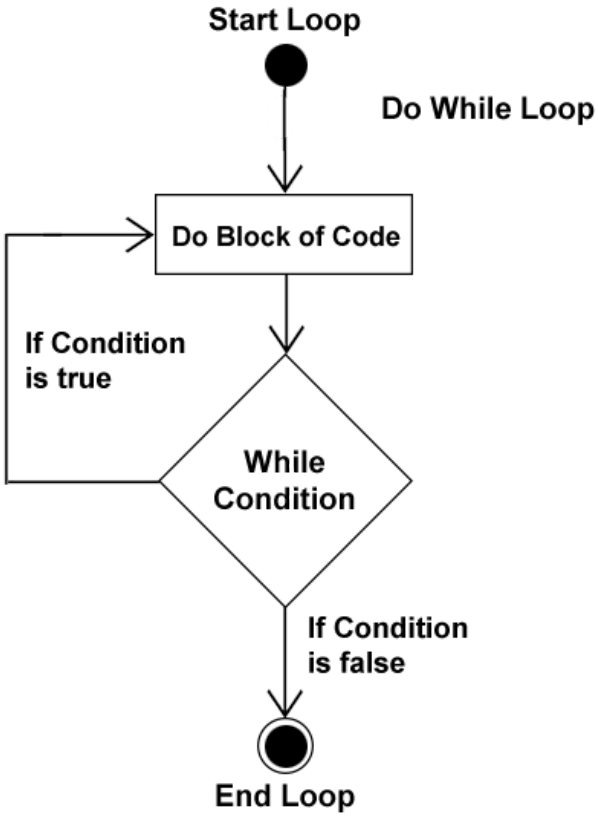
- For Loop को सिर्फ Variable Declaration कि जरुरत होती है | ये statement छोड़के सभी काम अपने अंदर ही करता है |
- जबतक While Loop में Condition true होती तब तक repeat होता रहता है |

Syntax :

```
for( variable_initialization; condition; increment/decrement){
statements;
}
```

For Example :

```
for( i=0; i<10; i++){
    printf("%d\n",i); // statement of while loop
 }
```



Source Code :

```
#include <stdio.h>
int main(){
   int i;
   for( i=0; i<10; i++){
    printf("%d\n",i); // statement of while loop
 }
return 0;
}
```

Output :

```
0
1
2
3
4
5
6
7
8
9
```

C Nested Loop

- Nested Loop ये loop का कोई प्रकार नहीं है |
- Nested Loop में एक loop में दूसरा loop लिया जाता है |
- Nested Loop While, Do-While, For Loop के होते है |

Source Code :

```
#include <stdio.h>
int main(){
int i,j;
for(i=1;i<4;i++){
    for(j=1;j<4;j++){
    printf("%d %d\n",i,j);
}
}
return 0;
}
```

Output :

```
1       1
1       2
1       3
2       1
2       2
2       3
3       1
3       2
3       3
```

Control Statements

C If Statement

- if Statement में अगर Condition true होती है तब Statement Execute होता है |

Syntax :

```
if(condition){
statement(s);
}
```

For Example :

```
int a=10, b=20;
if( a < b ){
printf("a is less than b");
}
```

Source Code :

```
#include <stdio.h>
int main(){
int a=10, b=20;
  if (a < b)
  {
  printf("a is less than b");
  }
return 0;
}
```

Output :

```
a is less than b
```

C If Else Statement

- if_else Statement में अगर Condition true हो तो वो if का statement Execute करता है |
- अगर Condition false होती है तो else का Condition करता है |

Syntax :

```
if(condition){
statement(s);
}else{
statement(s);
}
```

For Example :

```
int a=10, b=20;
if( a == b ){
printf("a is equal to b");
}else{
printf("a is not equal to b");
}
```

Source Code :

```
#include <stdio.h>
int main(){
int a=10, b=20;
if( a == b ){
printf("a is equal to b");
}else{
printf("a is not equal to b");
}
return 0;
}
```

Output :

```
a is not equal to b
```

C Else If Statement

- else_if Statement में अगर if की Condition true होती है तो if का statement execute होता है | अगर if का condition false होता है तो वो अगले condition पर जाकर check करता है | अगर वो condition true होता है तो वो उसका statement execute करता है | अगर कोई भी condition true नहीं होती तो वो else का statement execute करता है |

Syntax :

```
if(condition){
statement(s);
}else if(condition){
statement(s);
}else{
statement(s);
}
```

For Example :

```
int a=10, b=20;
if( a < b ){
printf("a is less than b");
}else if( a > b ){
printf("a is greater than b");
}else{
printf("a is equal to b");
}
```

Source Code :

```
#include <stdio.h>
int main(){
int a=10, b=20;
if( a < b ){
printf("a is less than b");
}else if( a > b ){
printf("a is greater than b");
}else{
printf("a is equal to b");
}
return 0;
}
```

Output :

```
a is less than b
```

C Switch Case Statement

- Switch case statement में expression होता है और उससे related कुछ cases होते है | जो case उस expression या declare किये हुए variable से match होती है तब वो output में print होता है | अगर कोई भी case expression से match नहीं होती तो वो default का statement output में print करेगा | आपको हर statement के बाद break लगाना पड़ता है, इसका मतलब वो उसके पहले का statement ही print करेगा | अगर आप break नहीं लगाते तो वो पहला और दूसरा ये दोनों statement को print करेगा | default case के बाद break नहीं लगाते |

Syntax :

```
switch (expression){
case value1 :
statement1 ;
break;
case value2 :
statement2 ;
break;
default :
statement3 ;
}
```

Source Code :

```
#include <stdio.h>
 int main(){
 char Day='D';
 switch(Day){
case 'A' :
printf("\nToday is Sunday");
break;
case 'B' :
printf("\nToday is Monday");
break;
case 'C' :
printf("\nToday is Tuesday");
break;
case 'D' :
case 'E' :
printf("\nToday is Wednesday");
break;
case 'F' :
printf("\nToday is Thurday");
break;
case 'G' :
printf("\nToday is Friday");
break;
case 'H' :
printf("\nToday is Saturday");
break;
default :
printf("\nDay is Not Found");
}
return 0;
 }
```

Output :

```
Today is Wednesday
```

C Break Statement

- Break Statement Program के loops और switch case के execution का काम किसी condition पर बंद कर देता है |

Syntax :

```
break;
```

Source Code :

```
#include <stdio.h>
int main()
{
int i=0;
    while ( i < 20 ){
    printf("value of i is %d\n", i);
    i++;
    if ( i == 10)
    break;
    }
return 0;
}
```

Output :

```
value of i is 0
value of i is 1
value of i is 2
value of i is 3
value of i is 4
value of i is 5
value of i is 6
value of i is 7
value of i is 8
value of i is 9
```

C Continue Statement

- Continue Statement Program के loops के condition के हिसाब से बीचवाले statements को skip कर देता है और बादवाले statement को execute करता है |

Syntax :

```
continue;
```

Source Code :

```
#include <stdio.h>
int main(){
   int i;
   for(i=0;i<10;i++)
   {
     if(i==7)
     {
    printf("Number %d is skipped.\n",i);
    continue;
     }
    printf("%d\n",i);
   }
return 0;
}
```

Output :

```
0
1
2
3
4
5
6
Number 7 is skipped.
8
9
```

C Goto Statement

Go to ये C Programming का statement है | इसमें labels का use किया जाता है |

Goto Statement के दो प्रकार होते है |

- Forward
- Backward

जब goto statement कुछ statement को छोड़कर उसके अगले statement को execute करता है , उसे Forward goto statement कहते है और किसी पिछले या execute हुए statement को फिरसे execute करने के लिए अपने पिछले label पर जाता है , उसे Backward goto statement कहते है |

Syntax for Forward and Backward goto Statement

Syntax For Forward:

```
goto label ;
statement ;
-----------
label ;
```

Syntax For Backward:

```
label ;
statement ;
----------
goto label ;
```

Source Code :

```
#include <stdio.h>
int main(){
int num1, num2, num3;
char ch;
yes : printf("Enter two values\n");
    scanf("%d%d",&num1, &num2);
num3 = num1 + num2 ;
    printf("Addition of %d and %d  is %d\n", num1, num2, num3);
    printf("\nDo you want to continue y(yes) or n(No)");
    scanf(" %c",&ch);
    if(ch=='y'){
    goto yes;
    }
    else if(ch=='n'){
    goto no;
    }
    else{
    printf("You entered other key");
    }
no : printf("Do you want to exit ?, Press Enter");
return 0;
}
```

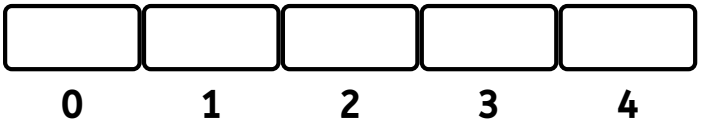
Output :

```
Enter two values
5
9
Addition of 5 and 9  is 14
Do you want to continue y(yes) or n(No)y
Enter two values
5
8
Addition of 5 and 8  is 13
Do you want to continue y(yes) or n(No)
```

Arrays

C Introduction of an Array

- Array ये same data type के variables का collection होता है |
- Array का data type कौनसा भी हो सकता है |
- Array के variables अपने पासवाले Memory Location पर store होते है
- |Array का initialization '0' से शुरू होता है |



Array का जो element पहला होता है उसका Memory Address सबसे कम होता है |

Array का इस्तेमाल क्यों किया जाता है ?

अगर किसी students के नाम store करना हो तो उनके अलग-अलग variable बनाने पड़ते है |
for eg.

```
char stud1 = "Rakesh" , stud2 = "Uday", stud3 = "Raj" ;
```

Array में इन सभी नाम के लिए सिर्फ एक ही array_variable create करना पड़ता है |
for eg.

```
char arr[] = { "Rakesh", "Uday", "Raj" };
```

C Single or One Dimensional Array

Syntax for Single Dimensional Array Declaration

```
data_type array_name[size_of_array];
```

For Example

```
int arr[5];
```

Syntax for Single Dimensional Array Initialization

```
data_type array_name[size_of_array] = {value1, value2,...,value n};
```

For Example

```
int arr[5] = {1, 2, 3, 4, 5};
```

Example for One/Single Dimensional Array

Source Code :

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

int i,num[5]={1, 2, 3, 4, 5};

for(i=0;i<5;i++){

printf("Array Number is %d\n",num[i]);
}
return 0;
}
```

Output :

```
Array Number is 1
Array Number is 2
Array Number is 3
Array Number is 4
Array Number is 5
```

Array Memory Representation

यहाँ पर Address में 4-4 का फर्क है क्योंकि, System 32-bit होने के कारण integer का size 4bytes है | इसीलिए Memory Address में 4-4 का फर्क है |

अगर array का data type character होता तो Address में 1-1 का फर्क होता |

index of array	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
array elements	1	2	3	4	5
Memory Address	6356728	6356732	6356736	6356740	6356744

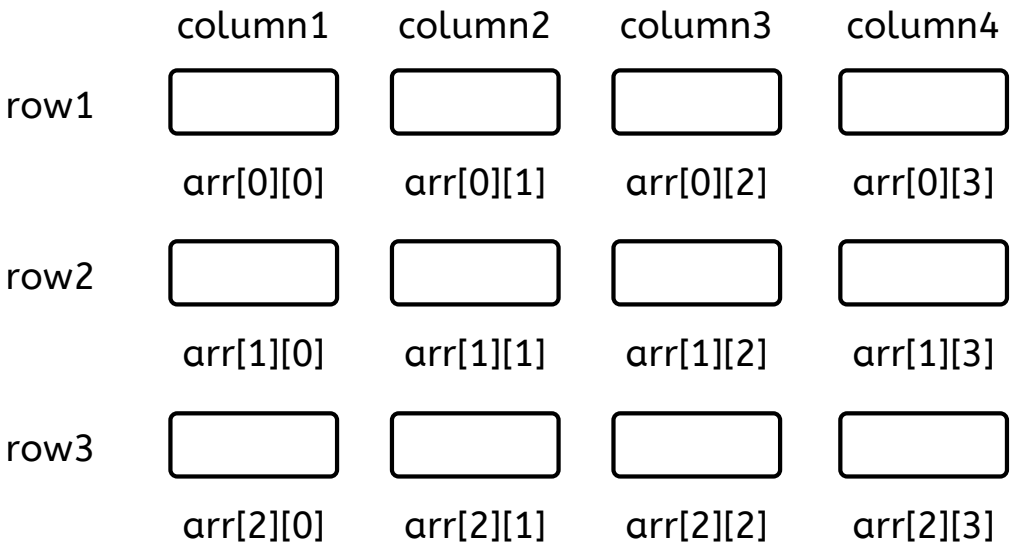
C Multi Dimensional Array

Syntax for Multi Dimentional Array Declaration

```
data_type array_name[ row_size ][ column_size ];
```

For Example

```
int arr[3][4];
```



Example for Multi Dimensional Array Initialization

```
int arr[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
OR
int arr[3][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

Example for Multi Dimensional Array

```
#include <stdio.h>
int main() {
int arr[3][4] = { {1,2,3,4}, {5,6,7,8}, {9,10,11,12} };
int i, j;
   for ( i = 0; i < 3; i++ ){
      for ( j = 0; j < 4; j++ ){
      printf("[%d][%d]=%d\n",i,j,arr[i][j]);
}
}
return 0;
}
```

Output :

```
[0][0]=1
[0][1]=2
[0][2]=3
[0][3]=4
[1][0]=5
[1][1]=6
[1][2]=7
[1][3]=8
[2][0]=9
[2][1]=10
[2][2]=11
[2][3]=12
```

Storage Classes

- Storage Classes Variables का scope और lifetime तय करता है |
- Storage Classes Variables को कहाँ पर store करके रखे ये बताता है | for eg. CPU, Register

C Automatic

- Automatic Storage Class में 'auto' keyword का इस्तेमाल करते है |
- ये Normal Variable की तरह ही होता है |
- ये एक Local Variable है |
- इनकी visibility या scope function के अंदर होता है |
- बाहर वो destroyed हो जाते है |इनकी default value 'garbage' होती है |

Syntax

```
auto data_type variable_name = value(optional);
```

For Example

```
int a ; // and
auto int a ; // both are same
```

Source Code :

```
#include <stdio.h>
int main(){
auto int a;
auto int b = 5;
printf("Value of a : %d\n", a);
printf("Value of b : %d", b);
return 0;
}
```

Output :

```
Value of a : 8 // garbage value
Value of b : 5
```

C External

- External Storage Class में 'extern' keyword का इस्तेमाल करते है |
- External Storage Class के variables का scope Global होता है |
- Global Variable के कारण इनका इस्तेमाल Program में extern के साथ कहा पर भी और किसी भी function के अंदर होता है |इनकी default value '0' होती है |

Syntax

```
extern data_type variable_name = value(optional);
```

For Example

```
extern int a ;
```

Source Code :

```
#include <stdio.h>
int num =  5 ;
void func();   // function declaration
int main(){
extern int num ;
    printf("Value of num : %d\n",num);
func();   // function calling
return 0;
}
void func(){     // function definition
extern int num ;
    printf("Value of num : %d",num);
}
```

Output :

```
Value of num : 5
Value of num : 5
```

C Register

- Register Storage Class में 'register' keyword का इस्तेमाल करते है |
- Register Storage Class के variables का scope Local होता है
- Local Variable के कारण इनका इस्तेमाल Program में जिस function के अंदर इनको declare या initialize किया है उसी function के अंदर visible रहता है |
- Register Storage Variables Computer के Register पर store होते है |
- Register Storage Class की Memory 'Limited' होती है | अगर Register की Memory ख़त्म हो जाए तो वो CPU Memory पर store होते है |
- Register Variables का कोई address(&) नहीं होता |
- इनकी default value 'garbage' होती है |

Syntax

```
register data_type variable_name = value(optional);
```

For Example

```
register int a ;
```

Source Code :

```
#include <stdio.h>
int main(){
register int num = 5 ;
    printf("Value of num : %d\n",num);
    // printf("Value of num : %d\n",&num);

return 0;
}
```

Output :

```
Value of num : 5
```

C Static

- Static Storage Class में 'static' keyword का इस्तेमाल करते है |
- Static Storage Class के Variables का scope Local और Global ये दोनों होता है |
- इनकी default value '0' होती है |

Code Description

निचे दिए हुए General File में int num = 1 ; ये value initialize की है और ये variable एक function में मतलब Local Variable लिया है | function1 इस function को for loop से repeat किया है | ये variable सामान्य होने के कारण इसकी value control के बाहर जाने के बाद नष्ट हो जाती है , इसीलिए इस variable की value increase नहीं होती बल्कि ये बार-बार initial होने से ये initial value को ही for loop के द्वारा बार-बार output में print करता है |

निचे दिए हुए Static File में static int num = 1 ; ये value initialize की है | इस program में General File के मुकाबले सिर्फ static keyword का use किया है | Static variables बिलकुल सामान्य variables के उलट प्रक्रिया करता है | Static Local variables की value control के बाहर जाने के बाद destroy नहीं होती | Static Local variables की value एक बार ही initialize होती है | Static Variables अपने control के बाहर जाने के बाद intialize हुई value को नष्ट नहीं करता, इसीलिए Static File में बार-बार initial value print नहीं होती |

General File :

Source Code :

```
#include <stdio.h>

func(){
int num = 1;
printf("%d",num);
num++;
}
int main()
{
int i;
for(i=0; i<5; i++){
printf("\n");
func();
}
return 0;
}
```

Output :

```
1
1
1
1
1
```

Static File :

Source Code :

```
#include <stdio.h>

func(){
static int num = 1;
printf("%d",num);
num++;
}
int main()
{
int i;
for(i=0; i<5; i++){
printf("\n");
func();
}
return 0;
}
```

Output :

```
1
2
3
4
5
```

Input and Output

C Scanf

- scanf ये एक standard library function है |
- scanf के जरिये numeric, characters और string कोई भी value input किया जाता है |
- scanf को program में इस्तेमाल करने के लिए stdio.h ये header file include करनी पडती है |
- scanf ये Keyboard से data read करने के लिए इस्तेमाल करते है |

Syntax

```
scanf("format_specifier(s)", &variable_list(s) );
```

निचे दिए हुए example में दो format specifiers और दो variables & (address operator or ampersand) के साथ लिए है | हर data types के अलग-अलग format specifiers होते है | यहाँ पर %d ये integer data type का format specifier है | data को Keyboard से read करने के लिए Variable के साथ '&' देना पड़ता है | अगर input देना हो तो Compiler को variable का address बताना पड़ता है |

```
scanf( "%d %d", &a, &b );
```

C Printf

- printf ये एक standard library function है |
- printf data को screen पर write करने के लिए इस्तेमाल किया जाता है |
- printf को program में इस्तेमाल करने के लिए stdio.h ये header file include करनी पडती है |

Syntax

```
printf("String / format_specifier(s) / escape sequence(s)", variable_list(s) );
```

For String

यहाँ पर सिर्फ string को इस्तेमाल किया गया है | String को Double quotes(" ") में लिखा जाता है और बाद में semi-colon देते है |

```
printf("Hello World !");
```

for format specifier and variable

ये scanf से input किये गए numeric (%d), character(%c), float(%f) या string(%s) output में print करता है |
यहाँ पर variable का address(&) नहीं दिया जाता |

```
printf("%d", a);
```

for escape sequence

यहाँ पर printf में \n(newline) ये escape sequence लिया है |

```
printf("%printf("\n");d", a);
```

Getchar And Putchar

1. getchar(Input) and putchar(Output) (for single character)

यहाँ पर सिर्फ string को इस्तेमाल getchar और putchar ये दोनों standard input/output (stdio.h) इस header file के library functions है,इसीलिए #include <stdio.h> ये preprocessor देना जरुरी है |
getchar का इस्तेमाल user से input मतलब एक character लेने की अनुमति ली जाती है |
अगर user ने एक से ज्यादा character input में दे दिए तो putchar पहले एक ही character को output में print करता है |किया गया है | String को Double quotes(" ") में लिखा जाता है और बाद में semi-colon देते है |

Source Code :

```
#include <stdio.h>
int main( ) {
int i;
printf( "Enter a character :");
i = getchar();
printf( "Entered character is : ");
putchar(i);
return 0;
}
```

Output :

```
Enter a character :hello
Entered character is : h
```

gets and puts

gets और puts ये दोनों standard input/output (stdio.h) इस header file के library functions है,इसीलिए #include <stdio.h> ये preprocessor देना जरुरी है |
gets का इस्तेमाल user से input मतलब string लेने की अनुमति ली जाती है | puts input लिए string को output में print करता है |

Source Code :

```
#include <stdio.h>
int main(){
char name[20];
printf( "Enter your name : \n");
gets(name);
printf( "You name is : ");
puts(name);
return 0;
}
```

Output :

```
Enter your name : Rakesh
Your name is : Rakesh
```

Functions

Function Introduction

- C Function ये statements का एक समूह होता है | हर एक Program में एक function तो होता ही है | for eg. main() Function को Program में कहा पर भी लिखा जाता है | जहाँ पर Function की जरुरत होती है वहाँ पर Function call किया जाता है |

Function के फायदे :

- Function में लिखा हुआ code बार-बार लिखना नहीं पड़ता |
- Function Programmer का समय और Program की space बचाता है |
- बड़े Program को छोटे-छोटे function में विभाजित किया जा सकता है |
- अगर Program में कहा पर error आ जाए तो उसे आसानी से निकाला जा सकता है |
- जहाँ पर जरुरत हो वहाँ पर function को बार-बार call किया जा सकता है |

Function कैसा होता है ?

Function का एक विशिष्ट नाम होता है | Function की शुरुआत में उसका नाम बाद में दो parethesis () और function के statements दो curly braces {} होते है |

Predefined or In built Function

- In-built Functions को predefined या Library Functions भी कहते है |
- In-built Functions में हर एक Functions के लिए अलग-अलग header file या preprocessor बनाये गए है |
- Function का declaration, definition header files में होता है |
- C में बहुत सारे Header files है इनमे अलग-अलग functions grouped करके रखे हुए है |अगर Programmer चाहे तो अपने खुदके header files भी बना सकता है |

For Example

printf() : ये Function stdio.h इस header file के अन्तर्गत आता है | अगर Programmer #include <stdio.h> ये preprocessor Program में include नहीं करता तो printf का इस्तेमाल नहीं कर सकता |

strlen() : ये Function string.h इस header file के अन्तर्गत आता है | अगर Programmer #include <string.h> ये preprocessor Program में include नहीं करता तो strlen का इस्तेमाल नहीं कर सकता |

User defined Function

User-defined Function के तीन विभाग होते है |

1. Function Declaration
2. Function Calling
3. Function Definition

1. In Function Declaration

Function create करते वक्त Compiler को बताना पड़ता है की आप कैसा Function create करना चाहते हो, इस process को Function Declaration कहते है |

Syntax for Function Declaration

```
return_type function_name(parameter(s));
```

Function Declaration के चार विभाग है |

1. return_type
2. function_name
3. parameter(s)/argument(s)
4. semicolon

1. return_type

हर Function कोई ना कोई value return करता है, वो null (void) हो या numeric(int, float, double) या character(char) | Function का return_type void default होता है |function का return_type program के requirement में होता है |

2. function_name

Function का नाम उसके code के अनुसार होना चाहिए | अगर ना भी हो तो भी compiler को चलता है, लेकिन ये Good Programming नहीं कहलाता | Function का नाम C का कोई keyword नहीं होता | C के Function के parenthesis () के अंदर parameters भी होते है |Function का नाम case-sensitive होता है | for eg. hello() और Hello() ये दोनों अलग-अलग functions है |

3. parameter(s)/argument(s)

Function के argument data_types होते है या data type के साथ उनके variable के नाम होते है | User किस प्रकार की value function calling के जरिये receive करना चाहता है, ये function के argument में declare होता है |

4. semicolon

हर Function के Declaration के बाद semicolon देते है | ये भी function declaration का हिस्सा है |

Example for Function Declaration with two parameters

```
int add(int a,int b);    // function declaration
```

Function Declaration without parameter(s)

```
int add();
```

Function Declaration with one parameter

```
int add(int a);
```

2. Function Calling

Syntax for Function Calling

```
function_name(Parameter1 ,Parameter2 ,.Parameter n);
```

Function Calling में सिर्फ function का नाम, function के arguments और semicolon होता है | निचे दिए हुए example function का return type 'integer' है | function का नाम 'add' है और function को दो parameters मतलब a और b पास किये गए है |जब तक function को call नहीं किया जाता तब तक function के declaration और definition को कोई महत्व नहीं रहता |

Example for Function calling with two parameters

```
add(a, b);  // funtion callling
```

Function calling without parameter(s)

```
add();
```

Function calling with one parameter

```
add( a );
```

3. Function Definition

Syntax for Function Definition

```
return_type function_name(Parameter(s)){

function_body;

}
```

Function Calling में सिर्फ function का नाम, function के arguments और semicolon होता है | निचे दिए हुए example function का return type 'integer' है | function का नाम 'add' है और function को दो parameters मतलब a और b पास किये गए है |जब तक function को call नहीं किया जाता तब तक function के declaration और definition को कोई महत्व नहीं रहता |

Function Definition के चार हिस्से

1. return_type

हर Function कोई ना कोई value return करता है, वो null (void) हो या numeric(int, float, double) या character(char) | Function का return_type void default होता है |function का return_type program के requirement में होता है |

2. function_name

Function का नाम उसके code के अनुसार होना चाहिए | अगर ना भी हो तो भी compiler को चलता है, लेकिन ये Good Programming नहीं कहलाता | Function का नाम C का कोई keyword नहीं होता | C के Function के parenthesis () के अंदर parameters भी होते है |Function का नाम case-sensitive होता है | for eg. hello() और Hello() ये दोनों अलग-अलग functions है |

3. parameter(s)/argument(s)

Function के argument data_types होते है या data type के साथ उनके variable के नाम होते है | User किस प्रकार की value function calling के जरिये receive करना चाहता है, ये function के argument में declare होता है |

4. function_body

Function body में variables होते है , लेकिन function के अंदर होने के कारण इनका scope Local होता है | Body के अंदर कुछ statement(s) होते है और value return करता है |

Example for Function Definition

```
int add(int x,int y){   // function definition
int z;
z = x + y ;
return z;
}
```

Full Example for Function

```
#include <stdio.h>
int add(int a,int b);  // function declaration with argument
int main(){
int a,b,c;
printf("Enter value of a and b : ");
scanf("%d %d", &a, &b);
c = add(a,b);   // funtion calling
printf("Addition of a and b : %d", c);
return 0;
}
int add(int x,int y){   // function definition
int z;
z = x + y ;
return z;
}
```

Call By Value And Reference

Call By Value और Call By Reference सिखने से पहले Parameters को समझे |

Function में दो प्रकार के Parameters होते है |

- Formal Parameter
- Actual Parameter

Formal Parameter

जो parameter function के declaration में और definition में लिखे जाते है, उसे Formal Parameter कहते है |

Full Example

```
void swap(int x, int y); // Formal Parameters
int main(){
int a=2; b=10
swap (a, b)
}
void swap (int x, int y){    //Formal Parameters
------------
------------
}
```

जो parameter function call में लिखे जाते है, उसे Actual Parameter कहते है |

Full Example

```
void swap(int x, int y);
int main(){
int a=2; b=10
swap (a, b)    //Actual Parameter
}
void swap (int x, int y){
------------
------------
}
```

Function Calling के दो प्रकार

1. Call By Value
2. Call By Reference

1. Call By Value

Call By Value में Variable के value को parameter के रूप से function को pass किया जाता है |
Call By Value में Actual Parameter की value Formal Parameter पर copy की जाती है |

यहाँ पर Program में variable 'a' और 'b' function 'add' को pass किये गए है |
यहाँ पर 'a' और 'b' की values 'x' और 'y' variable पर copy की जाती है |

For Example

```
#include <stdio.h>
void swap(int x, int y);
int main(){
int a = 2, b = 10;
printf("Before Swapping  a = %d and b = %d\n", a, b);
swap(a, b);
}
void swap(int x, int y)
{
int temp;
temp = x;
x = y;
y = temp;
printf("After Swapping a = %d and b = %d", x, y);
}
```

2. Call By Reference

Call By Reference में Variable के address को parameter के रूप से function को pass किया जाता है |
Call By Value में Actual Parameter की value Formal Parameter पर copy नहीं की जाती है |

यहाँ पर Program में variable 'a' और 'b' address को function 'add' को pass किये गए है |
यहाँ पर 'a' और 'b' की values 'x' और 'y' variable पर copy नहीं की जाती है |
ये सिर्फ variables का address hold करके रखता है |

For Example

```
#include <stdio.h>
void swap(int *x, int *y);
int main(){
int a = 2, b = 10;
printf("Before Swapping  a = %d and b = %d\n", a, b);
swap(&a, &b);
}
void swap(int *x, int *y)
{
int temp;
temp = *x;
*x = *y;
*y = temp;
printf("After Swapping a = %d and b = %d", *x, *y);
}
```

Pointers

Pointer Introduction

- हर एक Variable का एक Memory Address होता है और Memory Address देखने के लिए Programmer को variable से पहले '&'(address operator) लगाना पड़ता है |

For Example

```
int a ;
printf("Address of a : %d", &a);
```

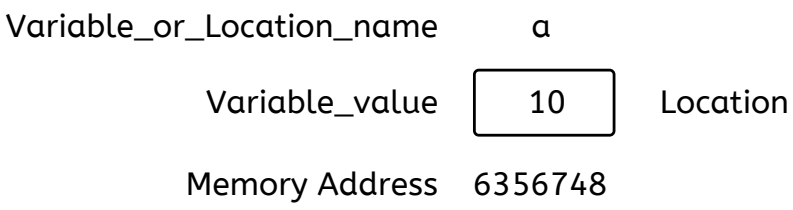
Output :

```
Address of a : 6356748    // 6356748 is address of variable
```

इसी memory address की मदद से Variable की value को access किया जाता है , इसी को 'Pointers' कहते है | Pointer किसी दूसरे variable का address hold करके रखता है |हर एक variable का एक address होता है | जब variable declare होता है तभी उसका एक memory location पर वो store होता है |

For Example

```
int a = 10;
```



Syntax for Pointer Declaration

```
data_type *pointer_name;
```

Example for Pointer Declaration

```
int *ptr1; // integer pointer variable
char *ptr2; // character pointer variable
float *ptr3; // float pointer variable
double *ptr4; // double pointer variable
```

किसी integer data type variable का address hold करना है तो pointer भी उसी data type का होगा जिस data type का variable हो, मतलब variable int है तो pointer भी int ही होगा, अगर variable char हो तो pointer variable भी char ही होगा | Pointer किसी की value hold करके नहीं रखता, बल्कि सिर्फ उसका address hold करके रखता है और उस address से ही pointer के साथ variable के value को print किया जाता है |

Full Example for Pointer

```
#include <stdio.h>
int main(){
int a = 10;
int *ptr;  // Integer Pointer Variable
ptr = &a;  //address of a stored in ptr
    printf("Value of a is: %d\n", a);
    printf("Address of a is: %u", ptr);
return 0;
}
```

Output :

```
Value of a is: 10
Address of a is: 6356744
```

Note : Pointer का address साधारणतः Hexadecimal Numbers होते है, पर कुछ compiler अपना address integer में print करते है | Variable का address print कराने के लिए '%x' या '%u' इन Format Specifiers का इस्तेमाल करते है |

Referencing Operator

Referencing मतलब किसी दूसरे variable का address hold करके रखना होता है |
हर variable का address hold करने के लिए जिसका address hold करना है और जिस Pointer variable में hold करना है, तो उन दोनों का data type same होना चाहिए |

For Example

```
int a = 10;
int *ptr;
ptr = &a;  //address of a stored in ptr
```

Dereferencing Operator

Dereferencing में asterisk (*) का इस्तेमाल करते पर pointer में इसे Dereferencing Operator भी कहते है | Dereferencing में pointer में store किये हुए value को access किया जाता है |

For Example

```
int a = 10;
int *ptr;
ptr = &a;
int b = *ptr; // ptr means dereferencing
printf("Value of a : %d", b);
```

Output :

```
Value of a : 10
```

Strings

What is String

- Strings characters का समूह होता है |
- Strings ये One-dimensional array होता है, जिसमे सिर्फ characters होते है |
- String का आखिरी character 'NULL'(\0) होता है |
- अगर पूरा string लिखना हो तो उसे double quotes (" ") में लिखा जाता है | अगर एक-एक character को लिखना हो तो उसे single quotes (' ') में लिखा जाता है |
- String का data type character (char) होता है |

Example for Single Character String

```
char str1[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

Example for Multiple Character String

```
char str2[6] = "Hello";
```

Example for Multiple Character String without using Array_size

```
char str3[] = "Hello";
```

Source Code :

```
#include <stdio.h>
int main(){
char str1[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
char str2[6] = "Hello";
char str3[] = "Hello";
    printf("Value of str1 : %s\n", str1);
    printf("Value of str2 : %s\n", str2);
    printf("Value of str3 : %s", str3);
return 0;
}
```

Output :

```
Value of str1 : Hello
Value of str2 : Hello
Value of str3 : Hello
```

String Representation

Note : String में एक से अधिक character होते है इसीलिए '%s' इस fomat specifier का इस्तेमाल करते है | अगर single character को print करना हो तो '%c' इस format specifier का इस्तेमाल किया जाता है |

Print String using '%s' format specifier

Note : String में एक से अधिक character होते है इसीलिए '%s' इस fomat specifier का इस्तेमाल करते है | अगर single character को print करना हो तो '%c' इस format specifier का इस्तेमाल किया जाता है |

```
char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
printf("Value of str : %s\n", str1);
```

Output :

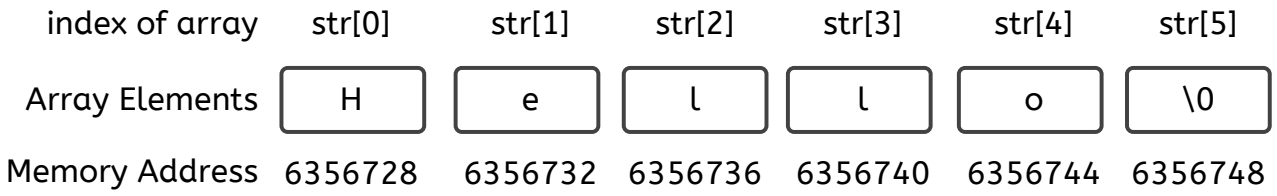
```
Value of str : Hello
```

Source Code :

```
char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
printf("Value of str[0] : %c\n", str[0]);
printf("Value of str[1] : %c\n", str[1]);
printf("Value of str[2] : %c\n", str[2]);
printf("Value of str[3] : %c\n", str[3]);
printf("Value of str[4] : %c\n", str[4]);
printf("Value of str[5] : %c\n", str[5]);
```

Output :

```
Value of str[0] : H
Value of str[1] : e
Value of str[2] : l
Value of str[3] : l
Value of str[4] : o
Value of str[5] : NULL Character is not shown
```



String Working with size (sizeof)

Program में हर एक String के initialization में अलग-अलग size है | दिए हुए array के size की memory allocate की जाती है | अगर Array का size नहीं दिया जाता तो जितनी size string की है उतनी size array allocate करता है |

Source Code :

```
#include <stdio.h>
int main(){
char str1[20] = {'H', 'e', 'l', 'l', 'o', '\0'};
char str2[10] = "Hello";  //using array_size
char str3[] = "Hello";    //without using array_size
    printf("Size of of str1 : %d\n", sizeof(str1));
    printf("Size of str2 : %d\n", sizeof(str2));
    printf("Size of str3 : %d", sizeof(str3));
return 0;
}
```

Output :

```
Size of of str1 : 20
Size of str2 : 10
Size of str3 : 6
```

String using *Pointer

Source Code :

```
#include <stdio.h>
int main(){
char *ptr = "Hello";
    printf("%s\n", ptr);
return 0;
}
```

Output :

```
Hello
```

String Library Functions

इन Functions को Program में इस्तेमाल करना हो तो string.h या strings.h इन header file को include करना पड़ता है |

String Functions	Description
strcat	एक String से दूसरे String को जोड़ा जाता है	
strchr	दिए हुए string से एक character का पहला occurrence के आगे का string pointer को return करता है	
strcmp	दो String को Compare किया जाता है	ये case-sensetive है	
strcmpi	दो String को Compare किया जाता है	ये case-sensetive नहीं है	
strcpy	एक string को दूसरे string में copy करता है	
strdup	String का duplicate बनाता है	
strlen	String की Length निकाली जाती है	
strlwr	Uppercase के Characters को Lowercase में convert किया जाता है	
strncat	दिए हुए number के जितने character है उनसे String को जोड़ा जाता है	
strncpy	दिए हुए number के जितने character एक string से दूसरे string में copy किया जाता है	
strnset	दिए हुए number और दिए हुए character के हिसाब से string को replace करता है	
strrchr	दिए हुए string से एक character का आखिरी occurrence के आगे का string pointer को return करता है	
strrev	String को उलटी दिशा से print करता है	
strrstr	दिए हुए String का आखिरी string occurrence के आगे का string pointer को return करता है	
strset	दिए हुए character से पूरे string को replace करता है	
strstr	दिए हुए String का पहला string occurrence के आगे का string pointer को return करता है	
strupr	Lowercase के Characters को Uppercase में convert किया जाता है	

strcat() - String Function

strcat ये String का एक Function है | एक String से दूसरे String को जोड़ा जाता है |

Syntax

```
strcat(destination_string, source_string);
```

- **Destination_string** - ये वो parameter है जिसके साथ source का string जोड़ा जाता है | String के आखिर में null character (\0) होता है | Source string destination string के साथ जुड़ते समय उसको remove कर देता है |
- **Source_string** - ये वो parameter है जिसका string destination string के साथ आखिर में जोड़ा जाता है |
किसी integer value को एक variable को दूसरे variable से जोड़ना हो तो arithmetic operator (+) का use नहीं कर सकते |

for e.g.
int num1 = 5 ;i
nt num2 = 5 ;
int num3 = num1 + num2 ;
इनका output 55 मिलना चाहिए लेकिन इनका output 10 मिलेगा |
किसी char को भी एक दूसरे से arithmetic operators के साथ नहीं जोड़ा जा सकता |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main (){
char str1[20] = "Welcome";
char str2[20] = " Friend";
strcat(str1, str2);
    printf("Concatenation String : %s", str1);
return 0;
}
```

Output :

```
Concatenation String : Welcome Friend
```

strchr() - String Function

दिए हुए string से एक character का पहला occurrence के आगे का string pointer को return करता है |

Syntax

```
strchr(string, int character);
```

- **string** - ये एक normal string है |
- **int character** - ये दिए हुए string में से दिए हुए character का पहला occurrence के आगे का string pointer को return करता है |

अगर दिया हुआ character string को नहीं मिलता तो वो NULL character return करता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main (){
char str[] = "Hello Friends";
char *ptr;
ptr = strchr(str, 'F');
    printf("%s", ptr);
return(0);
}
```

Output :

```
Friends
```

strcmp() - String Function

दो String को Compare किया जाता है | ये case-sensetive है |

Syntax

```
strcmp(string1, string2);
```

- **string1** -ये वो String है जिसके साथ String2 को Compare किया जाता है |
- **string2** - ये वो String है जिसके साथ String1 को Compare किया जाता है |

अगर दोनों string एक जैसे होते है तो ये '0' return करता है |अगर दोनों string अलग-अलग होते है तो '1' या '-1' return करता है |

Note : ये strcmp() function case-sensitive है | इसमें 'h' और 'H' ये दोनों अलग-अलग है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[] = "Hello" ;
char str2[] = "World" ;
int a = strcmp(str1, str2) ;
   printf ("%d\n", a) ;
int b = strcmp(str1, "Hello") ;
   printf ("%d\n", b) ;
int c = strcmp(str1, "hello") ; // strcmp is case-sensitive
   printf ("%d\n", c) ;

return 0;
}
```

Output :

```
-1
0
-1
```

strcmpi() - String Function

दो String को Compare किया जाता है | ये case-sensetive नहीं है |

Syntax

```
strcmpi(string1, string2);
```

- **string1** - ये वो String है जिसके साथ String2 को Compare किया जाता है |
- **string2** - ये वो String है जिसके साथ String1 को Compare किया जाता है |

अगर दोनों string एक जैसे होते है तो ये '0' return करता है |अगर दोनों string अलग-अलग होते है तो '1' या '-1' return करता है |

Note : ये strcmpi() function case-sensitive है | इसमें 'h' और 'H' ये एक ही अलग-अलग है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[] = "Hello" ;
char str2[] = "World" ;
int a = strcmpi(str1, str2) ;
   printf ("%d\n", a) ;
int b = strcmpi(str1, "Hello") ;
   printf ("%d\n", b) ;
int c = strcmpi(str1, "hello") ; // strcmpi is not case-sensitive
   printf ("%d\n", c) ;

return 0;
}
```

Output :

```
-1
0
0
```

strcpy() - String Function

दो String को Compare किया जाता है | ये case-sensetive नहीं है |

Syntax

```
strcpy(destination_string, source_string);
```

- **destination_string** - ये वो parameter है जिसपर source के string की value copy की जाती है | अगर destination string पर कोई value भी हो तो वो overwrite हो जाती है |
- **source_string** - ये वो parameter है जिसकी value destination पर copy की जाती है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[20] = "Welcome";
char str2[20] = "Friend";
    printf("Value of str2 = %s\n", str2 );
strcpy(str2, str1);
    printf("Copy str1 to str2 = %s", str2 );
return 0;
}
```

Output :

```
Value of str2 = Friend
Copy str1 to str2 = Welcome
```

strdup() - String Function

String का duplicate बनाता है |

Syntax

```
strdup(string);
```

- **string** - ये String है जिसका duplicate बनाया जाता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[] = "Hello World";
char *ptr;
ptr = strdup(str);
    printf("Duplicate string : %s", ptr);
return 0;
}
```

Output :

```
Duplicate string : Hello World
```

strlen() - String Function

String की Length निकाली जाती है |

Syntax

```
strlen(string);
```

- **string** - ये एक normal string है, जिसकी length निकली जाती है |

strlen से निकला हुआ output integer value ही होती है ,ये किसी दूसरे integer variable में भी store करके रख सकते है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[30] = "Hello World";
int length ;
   printf("String : %s\n", str);
length = strlen(str);
   printf("Length of str is %d", length);
return 0;
}
```

Output :

```
String : Hello World
Length of str is 11
```

strlwr() - String Function

Uppercase के Characters को Lowercase में convert किया जाता है |

Syntax

```
strlwr(string);
```

- **string** - ये वो string है जिसको lowercase में convert किया जाता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[50] = "HELLO WORLD";
int lwr ;
    printf("String :  %s\n", str );
    printf("Lowercase of str = %s", strlwr(str));
return 0;
}
```

Output :

```
String :  HELLO WORLD
Lowercase of str = hello world
```

strncat() - String Function

दिए हुए number के जितने character है उनसे String को जोड़ा जाता है |

Syntax

```
strncat(destination_string, source_string, size_t num);
```

- **destination_string** - ये वो string जिसके साथ source string को जोड़ा जाता है |
- **source_string** - ये वो string जिसके साथ destination string को बाद में जोड़ा जाता है |
- **size_t num** - यहाँ पर जो integer value दी जाती है उतने character वो source string से लेता है |

Program में strncat(str1, str2, 2); ऐसा लिखा है , इसका मतलब ये है कि, ये 2 characters str2 से लेगा |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[20] = "Hello";
char str2[20] = " World";
strncat(str1, str2, 2);
    printf("String :  %s\n", str1);
return 0;
}
```

Output :

```
String :  Hello W
```

strncpy() - String Function

दिए हुए number के जितने character एक string से दूसरे string में copy किया जाता है |

Syntax

```
strncpy(destination_string, source_string, size_t num);
```

- **destination_string** - ये वो parameter है जिसपर source के string की value copy की जाती है | अगर destination string पर कोई value भी हो तो वो overwrite हो जाती है |
- **source_string** - ये वो parameter है जिसकी value destination पर copy की जाती है |
- **size_t num** - यहाँ पर जो integer value दी जाती है उतने character वो destination string से लेकर source string पर copy कर देता है |

Program में strncpy(str1, str2, 2); ऐसा लिखा है , इसका मतलब ये है कि, ये 2 characters str2 से लेगा |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[20] = "Welcome";
char str2[20] = "Friend";
   printf("Value of str2 = %s\n", str2 );
strncpy(str2, str1, 2);
   printf("Copy str1 to str2 = %s", str2 );
return 0;
}
```

Output :

```
Value of str2 = Friend
Copy str1 to str2 = Weiend
```

strnset() - String Function

दिए हुए number और दिए हुए character के हिसाब से string को replace करता है |

Syntax

```
strnset(string, char ch, int c);
```

- **destination_string** - ये एक normal string है |
- **char ch** - ये वो character है जिससे string के हर character को replace किया जाता है |
- **int c** - यहाँ पर जितना number है उतने character string से replace किया जाते है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[20] = "Hello World";
   printf("Value of str = %s\n", str);
strnset(str, '.', 2);
   printf("%s", str);
return 0;
}
```

Output :

```
Value of str = Hello World
..llo World
```

strrchr() - String Function

दिए हुए string से एक character का आखिरी occurrence के आगे का string pointer को return करता है |

Syntax

```
strrchr(string, int character);
```

- **string** - ये एक normal string है |
- **int character** - ये वो character है जिसका आखिरी occurrence के आगे का string pointer को return किया जाता है |

अगर strrchr() function को कोई character नहीं मिलता तो वो NULL character return करता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[20] = "Hello World";
char *ptr;
ptr = strrchr(str, 'o');
   printf("%s", ptr);
return 0;
}
```

Output :

```
orld
```

strrev() - String Function

String को उलटी दिशा से print करता है |

Syntax

```
strrev(string);
```

- **string** - ये एक normal string है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[20] = "Hello World";
strrev(str);
    printf("%s", str);
return 0;
}
```

Output :

```
dlroW olleH
```

strrstr() - String Function

दिए हुए String का आखिरी string occurrence के आगे का string pointer को return करता है |

Syntax

```
strrstr(string1, string2);
```

- **string** - ये एक normal string है |
- **string2** - string1 में से ये string find करके उसका आखिरी occurrence pointer को return करता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[100] = "Hi How are you? Hi I am Fine";
char *ptr;
ptr = strrstr(str, "Hi");
    printf("%s", ptr);
return 0;
}
```

Output :

```
Hi I am Fine
```

strset() - String Function

दिए हुए String का आखिरी string occurrence के आगे का string pointer को return करता है |

Syntax

```
strset(string, int character);
```

- **string** - ये एक normal string है |
- **int character** - ये एक-एक character करके सभी characters को replace कर देता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[50] = "Hi How are you? Hi I am Fine";
strset(str, '$');
    printf("%s", str);
return 0;
}
```

Output :

```
$$$$$$$$$$$$$$$$$$$$
```

strstr() - String Function

दिए हुए String का आखिरी string occurrence के आगे का string pointer को return करता है |

Syntax

```
strstr(string1, string2);
```

- **string** - ये एक normal string है |
- **int character** - string1 में से ये string find करके उसका पहला occurrence pointer को return करता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[100] = "Hi How are you? Hi I am Fine";
char *ptr;
ptr = strstr(str, "Hi");
    printf("%s", ptr);
return 0;
}
```

Output :

```
Hi How are you? Hi I am Fine
```

strupr() - String Function

Lowercase के Characters को Uppercase में convert किया जाता है |

Syntax

```
strupr(string);
```

- **string** - ये वो string है जिसको Uppercase में convert किया जाता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[50] = "hello world";
int lwr ;
    printf("String :  %s\n", str );
    printf("Uppercase of str = %s", strupr(str));
return 0;
}
```

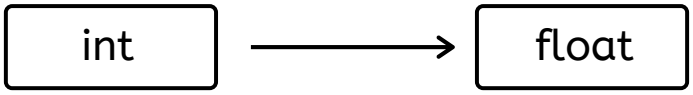
Output :

```
String :  hello world
Uppercase of str = HELLO WORLD
```

Type Casting

Type Casting Introduction

Type Casting में एक variable को दूसरे variable में convert किया जाता है |
For example अगर किसी variable को integer से float या float से integer में convert किया जाता है, तो उसे Type Casting कहते है |



Syntax

```
(data_type_name) expression
```

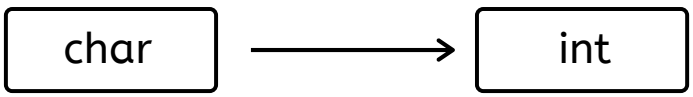
Source Code :

```
#include <stdio.h>
int main(){
int a, b;
float c;
    printf("Enter value of a : ");
    scanf("%d", &a);
    printf("Enter value of b : ");
    scanf("%d", &b);
   c = (float) a / b;
   printf("%f", c);
return 0;
}
```

Output :

```
Enter value of a : 15
Enter value of b : 6
2.500000
```

character to integer



Source Code :

```
#include <stdio.h>
int main(){
char c = 'a';
    printf("ASCII value of %c is %d", c, c);
return 0;
}
```

Output :

```
ASCII value of a is 97
```

Type Casting Functions

String Functions	Description
atoi()	character data type को int data type में convert किया जाता है ।
atof()	character data type को float-point(double) data type में convert किया जाता है ।
atol()	character data type को long data type में convert किया जाता है ।
itoa()	long data type को character data type में convert किया जाता है ।
ltoa()	long data type को character data type में convert किया जाता है ।

atoi() - Type Casting Function

इस Function को इस्तेमाल करना हो तो stdlib.h इस header file को include करना पड़ता है |
atoi() इस function से character data type को int data type में convert किया जाता है |

Syntax

```
int atoi(string);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
char str[10] = "0.01";
int i;
 i = atoi(str);
    printf("i = %d\n", i);
return 0;
}
```

Output :

```
i = 0
```

atof() - Type Casting Function

इस Function को इस्तेमाल करना हो तो stdlib.h इस header file को include करना पड़ता है |
atof() इस function से character data type को float-point(double) data type में convert किया जाता है |

Syntax

```
double atof(string);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
char str[10] = "0.01";
float i;
 i = atof(str);
    printf("i = %f\n", i);
return 0;
}
```

Output :

```
i = 0.010000
```

atol() - Type Casting Function

इस Function को इस्तेमाल करना हो तो stdlib.h इस header file को include करना पड़ता है |
atol() इस function से character data type को long data type में convert किया जाता है |

Syntax

```
long atol(string);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
char str[10] = "110000000000";
long i;
 i = atol(str);
    printf("i = %ld\n", i);
return 0;
}
```

Output :

```
i = 1100000000
```

itoa() - Type Casting Function

इस Function को इस्तेमाल करना हो तो stdlib.h इस header file को include करना पड़ता है |
itoa() इस function से long data type को character data type में convert किया जाता है |

Syntax

```
char itoa(long value, string, int base);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int a = 50;
char str[10];
long i;
itoa(a, str, 2);
   printf("Binary Number of %d is %s\n", a, str);
itoa(a, str, 8);
   printf("Octal Number of %d is %s\n", a, str);
itoa(a, str, 10);
   printf("Decimal Number of %d is %s\n", a, str);
itoa(a, str, 16);
   printf("Hexadecimal Number of %d is %s\n", a, str);
return 0;
}
```

Output :

```
Binary Number of 50 is 110010
Octal Number of 50 is 62
Decimal Number of 50 is 50
Hexadecimal Number of 50 is 32
```

ltoa() - Type Casting Function

इस Function को इस्तेमाल करना हो तो stdlib.h इस header file को include करना पड़ता है |
ltoa() इस function से long data type को character data type में convert किया जाता है |

Syntax

```
char ltoa(long value, string, int base);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int a = 50;
char str[10];
long i;
ltoa(a, str, 2);
   printf("Binary Number of %d is %s\n", a, str);
ltoa(a, str, 8);
   printf("Octal Number of %d is %s\n", a, str);
ltoa(a, str, 10);
   printf("Decimal Number of %d is %s\n", a, str);
ltoa(a, str, 16);
   printf("Hexadecimal Number of %d is %s\n", a, str);
return 0;
}
```

Output :

```
Binary Number of 50 is 110010
Octal Number of 50 is 62
Decimal Number of 50 is 50
Hexadecimal Number of 50 is 32
```

Structure

Introduction Of Structure

Structure ये एक अलग-अलग data types का collection होता है |
अगर structure का इस्तेमाल करना हो तो 'struct' keyword का इस्तेमाल करते है |
Structure ये array के जैसा ही होता है |
Array similar data types का collection होता है और Structure different data type का collection होता है |
Structure में किये हुए हर variable के decaration को 'member' कहते है |
Structure हर एक member के लिए अलग-अलग memory allocate करता है |

Syntax for Structure Definition

```
struct structure_name{
    data_type member 1;
    data_type member 2;
    ------------------
    ------------------
    data_type memeber n;
};
```

Example for Structure Definition :

```
struct Employee{
    int emp_id;
    char emp_name[30];
    float salary;
};
```

यहाँ पर example में structure का नाम 'Employee' लिया है और structure के तीन Members है जिनके अलग-अलग तीन data types है | एक 'emp_id' के लिए integer है, एक 'emp_name' के लिए character है, एक float है जो 'salary' के लिए है |

Structure Variable Declaration

Structure का variable declare करने के दो प्रकार है |

जब Structure की definition लिखी जाती तब वहा पर भी Structure variable को लिखा जाता है |

Structure के variable को main() function के अंदर भी किया जाता है |

Syntax for Structure Variable outside of Structure Definition

```
struct structure_name{
    data_type member 1;
    data_type member 2;
    ------------------
    ------------------
    data_type memeber n;
}structure_variable(s);
```

Example for Structure Variable in main() Function

```
struct Employee{
    int emp_id;
    char emp_name[30];
    float salary;
}info;
```

Syntax for Structure Variable in main() Function

```
struct structure_name{
    data_type member 1;
    data_type member 2;
    ------------------
    ------------------
    data_type memeber n;
}structure_variable(s);
int main(){
struct structure_name structure_variable_name;
}
```

Example for Structure Variable in main() Function

```
struct Employee{
    int emp_id;
    char emp_name[30];
    float salary;
};
int main(){
struct Employee info;
}
```

Structure के Members या Elements को access कैसे किया जाता है ?

Syntax for Accesing Structure members

```
structure_variable_name . member_of_structure = value(optional);
```

Example for Accesing Structure members

```
info.emp_id = 10;
```

Structure Initilization

Structure Initialization के दो प्रकार है |

Type 1 :

```
struct Employee{
    int emp_id;
    char emp_name[30];
    float salary;
};
int main(){
struct Employee info;
info.emp_id = 34;
    strcpy( info.emp_name, "Raj Biradar");
info.salary = 20000.00;
}
```

Type 2 :

```
struct Employee{
    int emp_id;
    char emp_name[30];
    float salary;
};
int main(){
struct Employee info = {34, "Raj Biradar", 20000.00};
}
```

Full Example for Structure

Source Code :

```
#include <stdio.h>
#include <string.h>
struct Employee {
   int  emp_id;
   char  emp_name[30];
   float  salary;
}info;
int main(){
   info.emp_id = 34;
   strcpy( info.emp_name, "Raj Biradar");
   info.salary = 20000.00;
   printf( "Employee id is : %d\n", info.emp_id);
   printf( "Employee name is %s\n", info.emp_name);
   printf( "Employee salary is : %f", info.salary);
return 0;
}
```

Output :

```
#include <stdio.h>
#include <string.h>
struct Employee {
   int  emp_id;
   char  emp_name[30];
   float  salary;
}info;
int main(){
   info.emp_id = 34;
   strcpy( info.emp_name, "Raj Biradar");
   info.salary = 20000.00;
   printf( "Employee id is : %d\n", info.emp_id);
   printf( "Employee name is %s\n", info.emp_name);
   printf( "Employee salary is : %f", info.salary);
return 0;
}
```

Structure working with size(sizeof)

Structure के member अलग-अलग memory allocate करता है और पूरा Structure का size उनके members के जितनी होती है |

Source Code :

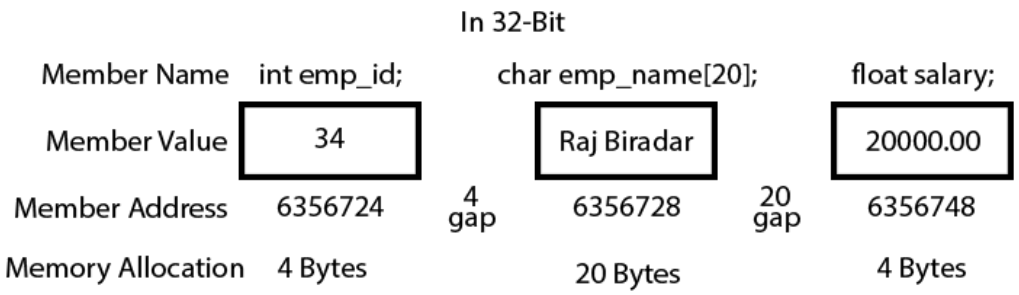
```
#include <stdio.h>
#include <string.h>
struct Employee {
    int emp_id;
    char emp_name[20];
    float salary;
};
int main(){
struct Employee info;
    printf( "Size of Employee id is : %d bytes\n", sizeof(info.emp_id));  // size of emp_id
    printf( "Size of Employee name : %d bytes\n", sizeof(info.emp_name)); // size of
emp_name
    printf( "Size of Employee salary is : %d bytes\n", sizeof(info.salary)); // size of salary
    printf( "Size of Employee structure : %d bytes", sizeof(info)); // size of Employee
return 0;
}
```

Output :

```
Size of Employee id is : 4 bytes
Size of Employee name : 20 bytes
Size of Employee salary is : 4 bytes
Size of Employee structure : 28 bytes
```

Structure Memory Representation

Memory Representation में देखे तो System 32-bit होने के कारण integer का size '4 Bytes' होता है | character का size bracket में दिए हुए size जितना ही होता है और float का size 4 Bytes ही होता है | Structure का size दिए हुए सभी Members के size जितना होता है | for eg. 4 (int) + 20 (char) + 4 (float) = 28 Bytes होता है |



Structure Using Pointer

Structure के Members को दो प्रकार से access किया जाता है |

- . (dot Operator)
- -> (pointer Operator)

Accessing Members using pointer Operator

Source Code :

```
#include <stdio.h>
#include <string.h>
struct Employee {
    int  emp_id;
    char  emp_name[30];
    float  salary;
};
int main(){
struct Employee info;
struct Employee *ptr;
ptr = &info;
    ptr->emp_id = 34;
    strcpy( ptr->emp_name, "Raj Biradar");
    ptr->salary = 20000.00;
    printf( "Employee id is : %d\n", ptr->emp_id);
    printf( "Employee name is %s\n", ptr->emp_name);
    printf( "Employee salary is : %f", ptr->salary);
return 0;
}
```

Output :

```
Employee id is : 34
Employee name is Raj Biradar
Employee salary is : 20000.000000
```

Structure Using Pointer

Syntax for Array Structure Declaration

```
struct structure_name structure_variable_name[array_size];
```

Example for Array Structure Declaration

```
struct Employee info[3];
```

अगर एक से ज्यादा Employee की information store करनी हो तो array का इस्तेमाल करता है |

Example for Structure Without using Array

Source Code :

```
#include <stdio.h>
#include <string.h>
struct Employee{
    int emp_id;
    char emp_name[20];
    float salary;
};
int main(){
    struct Employee info1, info2, info3;
    info1.emp_id=34;
    strcpy(info1.emp_name, "Raj");
    info1.salary = 20000.00;
    info2.emp_id=35;
    strcpy(info2.emp_name, "Maruti");
    info2.salary = 30000.00;
    info3.emp_id=36;
    strcpy(info3.emp_name, "Suraj");
    info3.salary = 40000.00;
    printf("Student 1\n");
    printf("Employee id is : %d\n", info1.emp_id);
    printf("Employee name is %s\n", info1.emp_name);
    printf("Employee salary is : %f\n\n", info1.salary);
    printf("Student 2\n");
    printf("Employee id is : %d\n", info2.emp_id);
    printf("Employee name is %s\n", info2.emp_name);
    printf("Employee salary is : %f\n\n", info2.salary);
    printf("Student 3\n");
    printf("Employee id is : %d\n", info3.emp_id);
    printf("Employee name is %s\n", info3.emp_name);
    printf("Employee salary is : %f\n\n", info3.salary);
return 0;
}
```

Ouput :

```
Student 1
Employee id is : 34
Employee name is Raj
Employee salary is : 20000.000000
Student 2
Employee id is : 35
Employee name is Maruti
Employee salary is : 30000.000000
Student 3
Employee id is : 36
Employee name is Suraj
Employee salary is : 40000.000000
```

Same Example for Structure using Array

Source Code :

```
#include <stdio.h>
#include <string.h>
struct Employee{
    int emp_id;
    char emp_name[20];
    float salary;
};
int main(){
    struct Employee info[3];
int i;
    info[0].emp_id=34;
    strcpy(info[0].emp_name, "Raj");
    info[0].salary = 20000.00;
    info[1].emp_id=35;
    strcpy(info[1].emp_name, "Maruti");
    info[1].salary = 30000.00;
    info[2].emp_id=36;
    strcpy(info[2].emp_name, "Suraj");
    info[2].salary = 40000.00;
for(i=0; i<3; i++){
    printf("Student %d\n", i+1);
    printf("Employee id is : %d\n", info[i].emp_id);
    printf("Employee name is %s\n", info[i].emp_name);
    printf("Employee salary is : %f\n\n", info[i].salary);
}
return 0;
}
```

Ouput :

```
Student 1
Employee id is : 34
Employee name is Raj
Employee salary is : 20000.000000
Student 2
Employee id is : 35
Employee name is Maruti
Employee salary is : 30000.000000
Student 3
Employee id is : 36
Employee name is Suraj
Employee salary is : 40000.000000
```

Union

What is Union

Union ये एक अलग-अलग data types का collection होता है |
अगर union का इस्तेमाल करना हो तो 'union' keyword का इस्तेमाल करते है |
Union ये structure के जैसा ही होता है |
Union में किये हुए हर variable के decaration को 'member' कहते है |
Union हर एक member के लिए अलग-अलग memory allocate नहीं करता है |
Union के members एक ही memory location को share करते है |
Union में जो member अपने size में बड़ा होता है, तो वो पूरे Union की size होती है |

Syntax Difference between Structure and Union Defintion

Syntax for Structure Definition

```
struct structure_name{
    data_type member 1;
    data_type member 2;
    ------------------
    ------------------
    data_type memeber n;
};
```

Syntax for Union Definition

```
struct union_name{
    data_type member 1;
    data_type member 2;
    ------------------
    ------------------
    data_type memeber n;
};
```

Syntax में सिर्फ struct और union keyword में फर्क है | Structure के members अलग-अलग memory allocate करते है | Union के members एक ही member की memory पर store होते है |

Example for Structure Definition

```
struct Employee{
    int emp_id;
    char emp_name[30];
};
```

Example for Union Definition

```
union Student{
    int stud_id;
    char stud_name[30];
};
```

Memory Representation

	Structure		Union		
Name	Employee		Student		
Member_name	emp_id	emp_name[30]	stud_id and stud_name[30]		Location
Memory Address	6356728	6356732	6356762		
Member Size	4bytes	30bytes	4bytes	30bytes	
Total Size	34bytes		30bytes		

Union Variable Declaration

Structure का variable declare करने के दो प्रकार है |
जब Union की definition लिखी जाती तब वहा पर भी Structure variable को लिखा जाता है |
Union के variable को main() function के अंदर भी किया जाता है |

Syntax for Union Variable outside of Union Definition

```
struct Union_name{
    data_type member 1;
    data_type member 2;
    ------------------
    ------------------
    data_type memeber n;
}Union_variable(s);
```

Example for Union Variable in main() Function

```
struct Employee{
    int emp_id;
    char emp_name[30];
    float salary;
}info;
```

Syntax for Union Variable in main() Function

```
union Union_name{
    data_type member 1;
    data_type member 2;
    ------------------
    ------------------
    data_type memeber n;
}Union_variable(s);
int main(){
union Union_name Union_variable_name;
}
```

Example for Union Variable in main() Function

```
union Employee{
    int emp_id;
    char emp_name[30];
    float salary;
};
int main(){
union Employee info;
}
```

Full Example for Union

Source Code :

```
#include <stdio.h>
#include <string.h>
union Employee {
   int emp_id;
   char emp_name[30];
   float salary;
}info;
int main(){
   info.emp_id = 34;
   strcpy( info.emp_name, "Raj Biradar");
   info.salary = 20000.00;
   printf( "Employee id is : %d\n", info.emp_id);
   printf( "Employee name is %s\n", info.emp_name);
   printf( "Employee salary is : %f", info.salary);
return 0;
}
```

Output :

```
Employee id is : 1184645120
Employee name is
Employee salary is : 20000.000000
```

Union working with size(sizeof)

Union के member एक ही memory location में store होता है और पूरे Union का size जो सबसे बड़ा member होता है, वो size Union की होती है |

Source Code :

```
#include <stdio.h>
#include <string.h>
union Employee {
   int emp_id;
   char emp_name[20];
   float salary;
};
int main(){
union Employee info;
   printf( "Size of Employee id is : %d bytes\n", sizeof(info.emp_id));  // size of emp_id
   printf( "Size of Employee name : %d bytes\n", sizeof(info.emp_name)); // size of
emp_name
   printf( "Size of Employee salary is : %d bytes\n", sizeof(info.salary)); // size of salary
   printf( "Size of Employee union : %d bytes", sizeof(info)); // size of Employee
return 0;
}
```

Output :

```
Size of Employee id is : 4 bytes
Size of Employee name : 20 bytes
Size of Employee salary is : 4 bytes
Size of Employee union : 20 bytes
```

Union using Pointer

Union के Members को दो प्रकार से access किया जाता है |

- . (dot Operator)
- -> (pointer Operator)

Accessing Members using pointer Operator

Source Code :

```
#include <stdio.h>
#include <string.h>
union Employee {
   int  emp_id;
   char  emp_name[30];
   float  salary;
};
int main(){
union Employee info;
union Employee *ptr;
ptr = &info;
   ptr->emp_id = 34;
   strcpy( ptr->emp_name, "Raj Biradar");
   ptr->salary = 20000.00;
   printf( "Employee id is : %d\n", ptr->emp_id);
   printf( "Employee name is %s\n", ptr->emp_name);
   printf( "Employee salary is : %f", ptr->salary);
return 0;
}
```

Output :

```
Employee id is : 1184645120
Employee name is
Employee salary is : 20000.000000
```

Typedef

Alias name for Data Type

Typedef ये C Programming के लिए बहुत ही महत्वपूर्ण keyword है | Typedef से मौजूदा data type को एक alias name(उपनाम) दिया जाता है | जिससे Programming करना और भी आसान हो जाता है |

Syntax

```
typedef data_type alias_name ;
```

Example for Typedef

```
typedef int myint ;
```

Example for Typedef Declaration

```
typedef int myint ;
myint a, b;
```

Normal Example for Typedef

Program में integer data type का alias name 'myint' लिया है | Program में integer जो काम करता है, वही काम 'myint' कर सकता है |
Note : अगर चाहे तो alias name का alias name भी programmer दे सकता है |

Source Code :

```
#include <stdio.h>
int main(){
typedef int myint;  //alias name of int is myint
myint a, b;
printf("Enter value of a : ");
scanf("%d", &a);
printf("Enter value of b : ");
scanf("%d", &b);
printf("Value of a : %d\n", a);
printf("Value of b : %d", b);
return 0;
}
```

Output :

```
Enter value of a : 4
Enter value of b : 5
Value of a : 4
Value of b : 5
```

Example for Typedef With Structure

Source Code :

```
#include <stdio.h>
typedef struct Employee{
     int emp_id;
     char emp_name[20];
     float salary;
 }info;
int main(){
    info Employee = {34, "Raj Biradar", 20000.00};
    printf("Employee id : %d\n",Employee.emp_id);
    printf("Employee name : %s\n",Employee.emp_name);
    printf("Employee salary : %.2f",Employee.salary);

return 0;
}
```

Output :

```
Employee id : 34
Employee name : Raj Biradar
Employee salary : 20000.00
```

Preprocessors

Preprocessors Introduction

Preprocessor क्या होता है ?

Program के हर header area में #include <stdio.h> इस प्रकार के files को include करते है | इनकी शुरुआत '#' से होती है | इन्हें 'Preprocessor' कहते है | Preprocessor को statement की तरह semicolon (;) नहीं दिया जाता |
Preprocessor ये एक Program है, जो Source Code compile होने से पहले ही execute होता है |

Normal Example for Preprocessor

Source Code :

```
#include <stdio.h>
int main(){
printf("Hello");
return 0;
}
```

यहाँ पर Preprocessors को अलग-अलग विभाग में विभाजित किया है |

Directive Types	Preprocessors
Macro	#define
File Inclusion	#include
Conditional Compilation	#if, #else, #ifdef, #endif, #ifndef, #elif
Other Directives	#pragma, #undef

Macro #define

macros identifiers होते है, Program में जहा पर identifier होता है, तो वो replace होकर वहा पर दी हुए value आ जाती है |

Source Code :

```
#include <stdio.h>
#define PI 3.145
int main(){
printf("Value of PI : %f\n", PI);
return 0;
}
```

Output :

```
Value of PI : 3.145000
Overwrite value of PI : 3.140000
```

Example for #define function macro

Source Code :

```
#include <stdio.h>
#define PI 3.145
#define Area(rad) PI*rad*rad
int main()
{
int radius = 5;
float area;
    area = Area(radius);
    printf("Area of circle is %f", area);
    return 0;
}
```

Output :

```
Area of circle is 78.625000
```

Predefined Macros

Macro	**Description**
DATE:	ये current date के string को return करता है	
FILE :	ये current file name को उसके path के साथ return करेगा	
LINE :	ये macro जिस line पर है उस line का number return करेगा	
STDC :	अगर compiler ANSI Standard C का पालन करता है तो वो '1' return करेगा	
TIME :	ये current time के string को return करता है	

Example for predefined macro

Source Code :

```
#include <stdio.h>
int main(){
    printf("Date : %s\n", __DATE__ );
    printf("File : %s\n", __FILE__ );
    printf("Line : %d\n", __LINE__ );
    printf("STDC : %d\n", __STDC__ );
    printf("Time : %s\n", __TIME__ );
return 0;
}
```

Output :

```
Date : Dec 19 2016
File : C:\UD\Documents\predefined_macro.c
Line : 7
STDC : 1
Time :16:38:22
```

File Inclusion #include

Header files को program में include करने के लिए #include का इस्तेमाल करते है |
हर एक program में #include का सन्दर्भ होता है |
Preprocessor दो प्रकार में include जाते है |

- #include
- #include "file_name"

कुछ header files predefined होते है और कुछ header files user-defined भी होती है |

- **Predefined header files** : stdio.h, conio.h, math.h
- **User-defined header file** : myheader.h

header files में functions होते है, जिस program में इन functions का इस्तेमाल करना हो तो header files को include करना पड़ता है |

- **stdio.h header files** : printf(), scanf()
- **myheader.h header file** : myfunction()

Source Code :

```
#include <stdio.h>
#include <conio.h>
void main(){
    printf("Hello World!");
getch();
}
```

Output :

```
Hello World!
```

Conditional Compilation

Conditional Compilation - #if

#if ये एक conditional preprocessor है , जिसके आगे expression या condition को लिखा जाता है | अगर दी हुई condition true हो तो वो नीचेवाला statement / code execute करता है |

Note :अगर Programmer CodeBlock के compiler में program बना रहा हो तो, उसे Program लिखते वक्त ही पता चलता है कि, कौनसा statement 'true' है और कौनसा 'false' |

Syntax for #if

```
#if condition
    statement(s);
#endif
```

Example

```
#include <stdio.h>
#define num 1
int main() {
#if (num==1)
{
printf("Number is equal to 1");
}
#endif
return 0;
}
```

Output :

```
Number is equal to 1
```

Conditional Compilation - #else

अगर #if की condition 'false" होती है तो #else का statement 'true' होता है |

Syntax for #else

```
#if condition
statement(s);
#else
statement(s);
#endif
```

Example

```
#include <stdio.h>
#define num 1
int main() {
#if (num==0)
{
    printf("Number is equal to 1");
}
#else
{
    printf("Number is equal to 1");
}
#endif
return 0;
}
```

Output :

```
Number is equal to 1
```

Conditional Compilation - #elif

#elif ये else if का combination होता है | अगर #if की condition false हो तो, वो #elif के condition पर जाता है , अगर condition true होती तो उसका statement print होता है |

Syntax for #elif

```
#if condition
statement(s);
#elif condition
statement(s);
#else
statement(s);
#endif
```

Example

```
#include <stdio.h>
#define num 15
int main(){
#if (num<10)
{
printf("Number is less than 10");
}
#elif (num>10)
{
printf("Number is greater than 10");
}
#else#if (num==10)
{
printf("Number is equal to 10");
}
#endif
return 0;
}
```

Output :

```
Number is greater than 10
```

Conditional Compilation - #ifdef

#ifdef(if defined) पहले program में #define ये preprocessor लिखा है या नहीं ये check करता है | अगर #ifdef का macro और #define का macro same होता है तो, #ifdef का statement 'true' होता है |

Syntax for #ifdef

```
#ifdef macro
statement(s);
#endif
```

Example

```
#include <stdio.h>
#define POSITIVE 5
int main() {
#ifdef POSITIVE
    printf("Defined Positive value");
#else
    printf("Defined Negative Value");
#endif
return 0;
}
```

Output :

```
Defined Positive value
```

Conditional Compilation - #ifndef

#ifndef (if not defined) ये पहले #define के macro को check करता है, अगर #define का macro और #ifndef का macro अलग-अलग है तो, वो true return करता है |

Syntax for #ifndef

```
#ifndef macro
statement(s);
#endif
```

Example

```
#include <stdio.h>
#define POSITIVE 5
int main() {
#ifndef NEGATIVE
    printf("Defined Negative value");
#endif
return 0;
}
```

Output :

```
Defined Negative value
```

Conditional Compilation - #endif

जहा पर #if का इस्तेमाल होता है वहा पर #endif का इस्तेमाल होता है |

Syntax for #ifndef

```
statement(s) of #if or #else;
#endif
```

Example

```
#include <stdio.h>
#define num 1
int main() {
#if (num==1)
{
printf("Number is equal to 1");
}
#endif
return 0;
}
```

Output :

```
Number is equal to 1
```

Other Directives #undef

Conditional Compilation - #endif

#undef ये preprocessor #define किये हुए macro को undefine कर देता है |

Syntax for ##undef

```
#undef macro
```

Example 1

```
#include <stdio.h>
#define num 1
#undef num
int main(){
    printf("num is %d\n", num);
return 0;
}
```

Output :

```
error: 'num' undeclared
```

Example 2

```
#include <stdio.h>
#define num 1
int main(){
printf("num is %d\n", num);
#undef num
#define num 2
printf("num is %d", num);
return 0;
}
```

Output :

```
num is 1
num is 2
```

File Handling And Functions

File Handling में data को secondary storage device(Hard disk) में permanently store किया जाता है |
File Handling को open, close, read, write के लिए इस्तेमाल किया जाता है |

File Handling का इस्तेमाल क्यों करते है ?

Program जब Programmer बंद करता है, तब program का सारा data destroy हो जाता है | Program में data store करने के Programmer कुछ variables, arrays, structures, unions का इस्तेमाल करता है, पर ये data permanently store नहीं होता | इसे permanently store करने के लिए ही File Handling का इस्तेमाल होता है | File Handling के दरम्यान create हुई files चाहे वो अलग-अलग types(.txt, .doc etc.) की हो वो portable होती है | दूसरे Computer में भी वो इस्तेमाल होती है |

File Handling के Operations

File Handling के Functions को इस्तेमाल करना हो तो stdio.h इस header file को include करना पड़ता है |

File Operations / Functions	Description
fopen()	File को create और open किया जाता है	
fclose()	Open किये हुई file को close किया जाता है	
fprintf()	File में data को write किया जाता है	
fscanf()	File में data को read किया जाता है	
fgetc()	File से character को read किया जाता है	
fputc()	File में character को write किया जाता है	
fseek()	दिए हुई position पर file pointer को set किया जाता है	
getw()	File से integer को read किया जाता है	
putw()	File में integer को write किया जाता है	
ftell()	File Pointer की current position return करता है	
rewind()	File Pointer को file के शुरुआत में लाता है	
remove()	File को remove कर देता है	

File को open करने से पहले उनके modes को समझे |

File को open करने के लिए कुछ 'modes' होते है |

File Modes	Description
r	read mode पे file को open किया जाता है	
w	write mode पे file को open किया जाता है	
a	append mode मतलब file में कुछ और data जोड़ने के file को open किया जाता है	
r+ w+ a+	read और write modes पे file को open किया जाता है	
rb	read mode पे binary file को open किया जाता है	
wb	write mode पे binary file को open किया जाता है	
ab	append mode मतलब binary file में कुछ और data 'write' जोड़ने के binary file को open किया जाता है	
rb+ wb+ ab+	read और write modes पे binary file को open किया जाता है	

fopen() - File Handling

fopen() funtion से File को create और open किया जाता है |

Syntax

```
file_pointer = fopen("file_name", "mode_of_file");
```

file_pointer : ये एक file pointer है जिसका data type 'FILE' है |
file_name : File का नाम उसके extension के साथ देना चाहिए |for e.g. file.txt, file.docx .txt और .docx ये files के extensions है |
mode_of_file : File को किस mode से open करना ये उनका mode निश्चित करता है | File को Open करने के तीन basic modes है |

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
fptr = fopen("sample.txt","w");
return 0;
}
```

fclose() - File Handling

fclose() funtion से open किये हुए file को close किया जाता है |

Syntax

```
fclose(file_pointer);
```

file_pointer : जिस file को close करना है उस file का pointer |

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
fptr = fopen("sample.txt","w");
fclose(fptr);
return 0;
}
```

fprintf() - File Handling

fprintf() funtion से File में data को write किया जाता है |

Syntax

```
fprintf(file_pointer, "format_specifier or string", variables);
```

file_pointer : जिस file को write करना है उस file का pointer |
format specifier or string: जैसे printf में इस्तेमाल किया जाता है, वैसे ही यहाँ format specifier और string का इस्तेमाल यहाँ पर होता है | दोनों एक साथ भी इस्तेमाल किये जा सकते है |
variables : अगर format specifier जहाँ पर दिया जाता है, वह पर variable तो होते ही है | अगर format specifier नहीं होते तो variables भी नहीं होते |

Source Code :

```
#include <stdio.h>
int main(){
int emp_id;
char emp_name[20];
FILE *fptr;
fptr = fopen( "sample.txt", "w" );
printf("Enter Employee ID :");
scanf("%d", &emp_id);
printf("Enter Employee Name :");
scanf("%s", &emp_name);
fprintf( fptr, "%d, %s", emp_id, emp_name ) ;
fclose(fptr);
return 0;
}
```

Output :

```
Enter Employee ID :2
Enter Employee Name :Raj
```

fscanf() - File Handling

fscanf() funtion से File में data को read किया जाता है |

Syntax

```
fscanf(file_pointer, "format_specifier or string", &variables);
```

file_pointer : जिस file को read करना है उस file का pointer |
format specifier or string: जैसे scanf में इस्तेमाल किया जाता है, वैसे ही यहाँ format specifier और string का इस्तेमाल यहाँ पर होता है | दोनों एक साथ भी इस्तेमाल किये जा सकते है |
variables : अगर format specifier जहाँ पर दिया जाता है, वह पर variable तो होते ही है | अगर format specifier नहीं होते तो variables भी नहीं होते |

Source Code :

```
#include <stdio.h>
int main(){
int emp_id;
char emp_name[20];
FILE *fptr;
fptr = fopen( "sample.txt", "r" );
fscanf( fptr, "%d,%s", &emp_id, emp_name ) ;
    printf("Employee ID : %d\n", emp_id);
    printf("Employee Name : %s", emp_name);
fclose(fptr);
return 0;
}
```

Output :

```
Employee ID : 2
Employee Name : Raj
```

fgetc() - File Handling

fgetc() funtion से File में data को read किया जाता है |

Syntax

```
fgetc(file_pointer);
```

file_pointer : जिस file को read करना है उस file का pointer |

ये normally file में से single character ही return करता है | पर feof (end of file) होने के कारण जबतक file में से characters ख़त्म नहीं होते तबतक ये one-by-one characters print करते रहता है |

Source Code :

```
#include <stdio.h>
int main(){
char ch;
FILE *fptr;
fptr = fopen("sample.txt","r");
while((!feof(fptr))){

ch = fgetc(fptr);
    printf("%c",ch);
}
fclose(fptr);
return 0;
}
```

Output :

```
2, Raj
```

fputc() - File Handling

fputc() funtion से File में data को write किया जाता है |

Syntax

```
fputc(character, file_pointer);
```

character : जो character file में write करना है वो character यहाँ पे आता है |
file_pointer : जिस file को write करना है उस file का pointer |

Source Code :

```
#include <stdio.h>
int main(){
char ch;
FILE *fptr;
fptr = fopen("sample.txt","w");
    fputc('h',fptr);
fclose(fptr);
return 0;
}
```

Output :

```
h
```

fseek() - File Handling

fseek() funtion से दिए हुई position पर file pointer को set किया जाता है |

Syntax

```
fseek(file_pointer, offset, position);
```

file_pointer : जिस file को read और write करना है उस file का pointer |
offset : File के data को beginning(SEEK_SET) से, end(SEEK_END) से और current(SEEK_CUR) position से कितने characters या bytes replace या store करना है वो integer value यहाँ पर आती है |
position : File के data को कहा से replace करे या store करे ये वो position है |for eg. SEEK_SET(0), SEEK_CUR(1), SEEK_END(2)

sample.txt

```
Hello World
```

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
char str[20];
fptr = fopen("sample.txt","r+");
fgets(str, 20, fptr);
    printf("Old file contents : %s\n", str);   // Old contents : Hello World
fseek(fptr, 6, SEEK_SET);   // replace contents after 6 characters
fputs("Friends", fptr);     // put Friends after 6 characters
fclose(fptr);    //file is closed
fptr = fopen("sample.txt","r");   // file is reopened
                                  // User can use rewind() function instead of reopening file
fgets(str, 20, fptr);
    printf("New file contents : %s\n", str);
fclose(fptr);
return 0;
}
```

Output :

```
Old file contents : Hello World
New file contents : Hello Friends
```

getw() and putw() - File Handling

putw()

putw() funtion से Integer value को file में write किया जाता है |

Syntax

```
putw(integer, file_pointer);
```

integer : जो integer value या उसका variable file में write करना है वो यहाँ पर आता है |
file_pointer : जिस file में integer value को write करना है वो file pointer यहाँ पर आता है |

getw()

getw() funtion से Integer value को file में से read किया जाता है |

Syntax

```
getw(file_pointer);
```

file_pointer : जिस file का integer read करना है उस file का pointer |

Source Code :

```
#include <stdio.h>
int main()
{
FILE *fptr;
int num = 5;
fptr=fopen("file.txt","w");
putw(num,fptr);
    printf("put value : %d\n",num);
fclose(fptr);
fptr=fopen("file.txt","r");  // file is reopened
int num1 = getw(fptr);
    printf("get value : %d\n",num1);
fclose(fptr);
return 0;
}
```

Output :

```
put value : 5
get value : 5
```

ftell() - File Handling

ftell() funtion File Pointer की current position return करता है |

Syntax

```
ftell(file_pointer);
```

file_pointer : ये एक file pointer है | जिससे file की current position समझ आएगी |

ये long integer को return करता है | इससे file के data को bytes में नापता है | अगर कुछ कारन से file open होने में दिक्कत आती है, तो ये -1 return करता है |

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
char str[] = "Hello World";
fptr=fopen("file.txt","w");
fputs(str,fptr);
    printf("File Contents : %s\n",str);
int size = ftell(fptr);
    printf("Size of file in bytes : %d\n", size);
fclose(fptr);
return 0;
}
```

Output :

```
File Contents : Hello World
Size of file in bytes : 11
```

rewind() - File Handling

rewind() funtion से file pointer को file के शुरुआत में लाता है |

Syntax

```
rewind(file_pointer);
```

file **pointer** : जिस file को इस file pointer में open किया है, उस file pointer को शुरुआत में लाता है |

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
char str[20];
fptr = fopen("file.txt","r+");
fgets(str, 20, fptr);
    printf("Old file contents : %s\n", str);
fseek(fptr, 6, SEEK_SET);
fputs("Friends", fptr);
rewind(fptr);   // file is reopened
fgets(str, 20, fptr);
    printf("New file contents : %s\n", str);
fclose(fptr);
return 0;
}
```

Output :

```
Old File Contents : Hello World
New file contents : Hello Friends
```

remove() - File Handling

remove() funtion से file को delete या remove किया जाता है |

Syntax

```
remove(file_name);
```

file_name : जिस file को remove करना उस file का नाम यहाँ पर आएगा |

जिस directory पर programmer निचे दिया हुआ source code save करता है, उस directory की file ही delete की जाती है | अगर file को delete करना हो तो उस file को उसके extension/type के साथ देना पड़ता है | file को delete करते वक्त सोचे कि ये file permanantly delete हो जायेगी | यहाँ से delete होने वाली file 'Recycle Bin' में नहीं जाती |

Source Code :

```
#include <stdio.h>
int main(){
int remover;
char file[20];
   printf("Enter file name to delete : ");
   scanf("%s", file);
remover = remove(file);
if(remover == 0){
   printf("File is removed.");
}else{
   printf("Unable to delete file.");
}
return 0;
}
```

Output :

```
Enter file name to delete : hello.c
Unable to delete file.
```

Bit Fields

- Bit Fields ये Program की memory को बचाता है |
- Bit Fields Structure और Union में काम करता है |

Syntax for Bit Fields

```
struct_or_union structure_or_union_name{
   data_type member 1: bit(s);
   data_type member 2: bit(s);
   data_type member n: bit(s);
}structure_or_union variable;
```

Normal Program for Structure

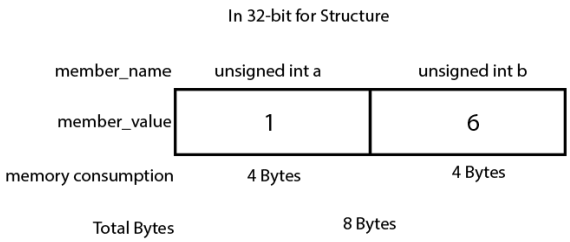
अबतक Normal Structure के Program में Bit Field का इस्तेमाल नहीं किया है | इस Program में structure ने 8 bytes की memory consume की है |

Source Code :

```
#include <stdio.h>
struct Numbers{
unsigned int a;
unsigned int b;
}value;
int main() {
value.a = 1;
value.b = 6;
printf("Size of a : %d bytes\n", sizeof(value.a));
printf("Size of b : %d bytes", sizeof(value.b));
return 0;
}
```

Output :

```
Size of a : 4 bytes
Size of b : 4 bytes
Size of Numbers structure : 8 bytes
```



Program for Structure using Bit Fields

निचे दिए हुए Program ने अपना bit field रखा है | member 'a' के लिए सिर्फ 1 bit लिया है और 'b' के लिए 3 bits लिए है | पूरा structure सिर्फ 4 bits ही इस्तेमाल करेगा | लेकिन Program के output में structure ने 4 bytes का इस्तेमाल किया है | हर 1 byte = 8 bits का होता है और यहाँ पर 4 bytes = 32 bits structure ने इस्तेमाल कर ली है | Structure कम से कम 4 bytes ही consume कर लेता है , इससे कम memory consume नहीं की जाती | अगर Programmer चाहे तो और भी members declare कर सकता है |

for eg.

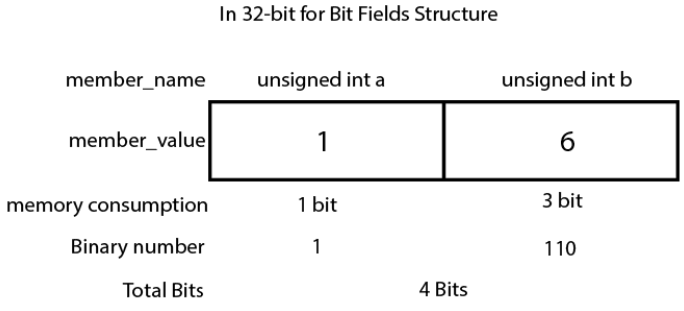
- 1 bits के 32 members Programmer 4 bytes में declare कर सकता है |
- 2 bits के 16 members Programmer 4 bytes में declare कर सकता है |
- 3 bits के 30 members Programmer 4 bytes में declare कर सकता है |
- 4 bits के 8 members Programmer 4 bytes में declare कर सकता है |
- 8 bits के 4 members Programmer 4 bytes में declare कर सकता है |
- 16 bits के 2 members Programmer 4 bytes में declare कर सकता है |
- 32 bits के 1 members Programmer 4 bytes में declare कर सकता है |

Source Code :

```
#include <stdio.h>
struct Numbers{
unsigned int a : 1;
unsigned int b : 3;
}value;
int main() {
value.a = 1;
value.b = 6;
printf("Size of structure Numbers : %d bytes\n", sizeof(value));
return 0;
}
```

Output :

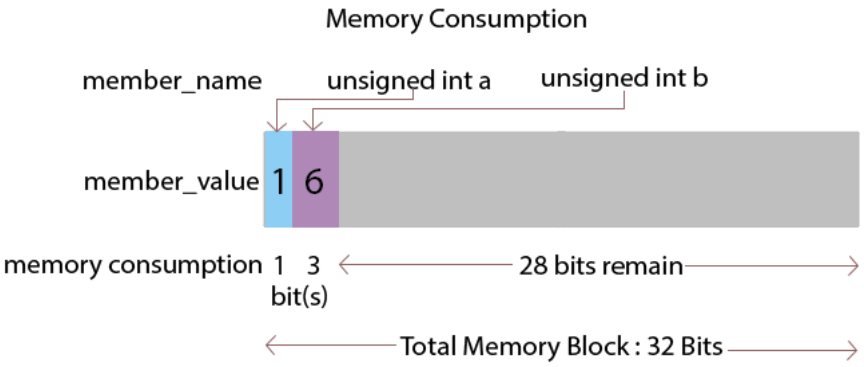
```
Size of Numbers structure : 4 bytes
```



Bits की संख्या कैसे दे ?

Bits binary numbers पर काम करता है | अगर कोई integer value को देना हो तो सोचना चाहिए कि उसे कितने bits लगेंगे for eg. अगर यहाँ पर integer की value '6' ले ली तो 4 का binary '110' है मतलब binary की संख्या '3' है इसीलिए integer की value '6' के लिए 3 bits को set करे |

Bit Fields के साथ Structure Bits कैसे लेता है ?



Source Code :

```
#include <stdio.h>
struct Numbers{
unsigned int a : 1;
unsigned int b : 3;
}value;
int main() {
value.a = 1;
value.b = 6;
printf("Size of structure Numbers : %d bytes\n", sizeof(value));
return 0;
}
```

Output :

```
Size of Numbers structure : 4 bytes
```

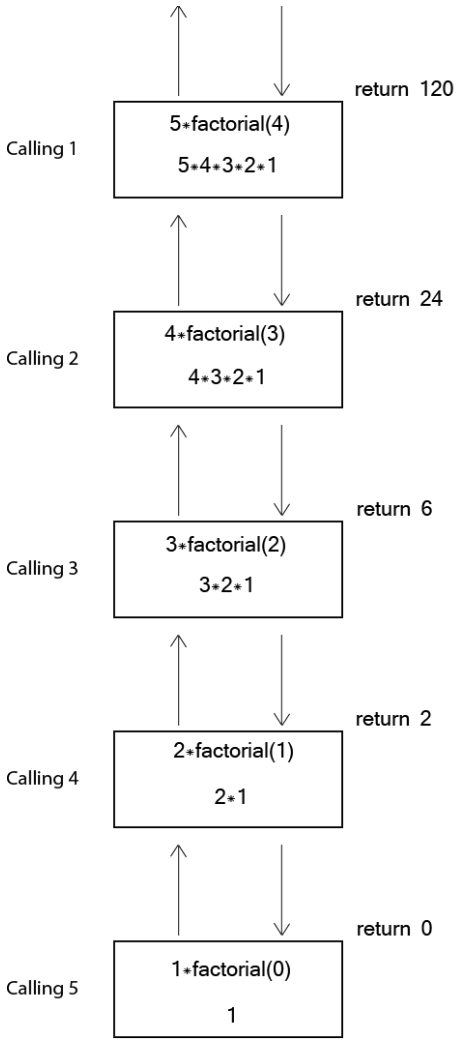
Recursion

- जिस function में वही function को call किया जाता है, उसे recursion कहते है |
- Recursion एक ऐसा process है , जो loop के तरह काम करता है |
- Recursion को एक satisfied condition लगती है, जिससे recursive function काम करना बंद कर दे |
- Recursive Function तबतक call होता रहता है, जबतक उसका satisfaction नहीं होता |
- अगर Recursive function का satisfaction नहीं हो तो 'infinite looping' की संभावना होती है |

Recursive function कैसा होता है ?

Syntax

```
void recursive_function(){
//some code;
   recursive_function()();
//some code;
}
int main(){
//some code;
   recursive_function()();
//some code;
}
```



Example

```
#include <stdio.h>
int main(){
int a,fact=0;
   printf("Enter Number : ");
   scanf("%d", &a);
fact=factorial(a);
   printf("Factorial of %d is %d", a, fact);
return 0;
}
int factorial (int n)
{
if (n == 0){
   return 1;
}else{
   return (n * factorial(n-1));
}
}
```

Output :

```
Enter Number : 4
Factorial of 4 is 24
```

Example for Addition of Two Numbers using Recursion

Source Code :

```
#include <stdio.h>
int add(int, int);
int main()
{
int num1,num2,sum;
    printf("Enter Two Values\n");
    scanf("%d %d",&num1,&num2);
sum = add(num1, num2);
    printf("sum of %d and %d : %d", num1, num2, sum);
return 0;
}
add(int a, int b){
if(b==0){
    return a;
}else{
    return (1+add(a,b-1));
}
}
```

Output :

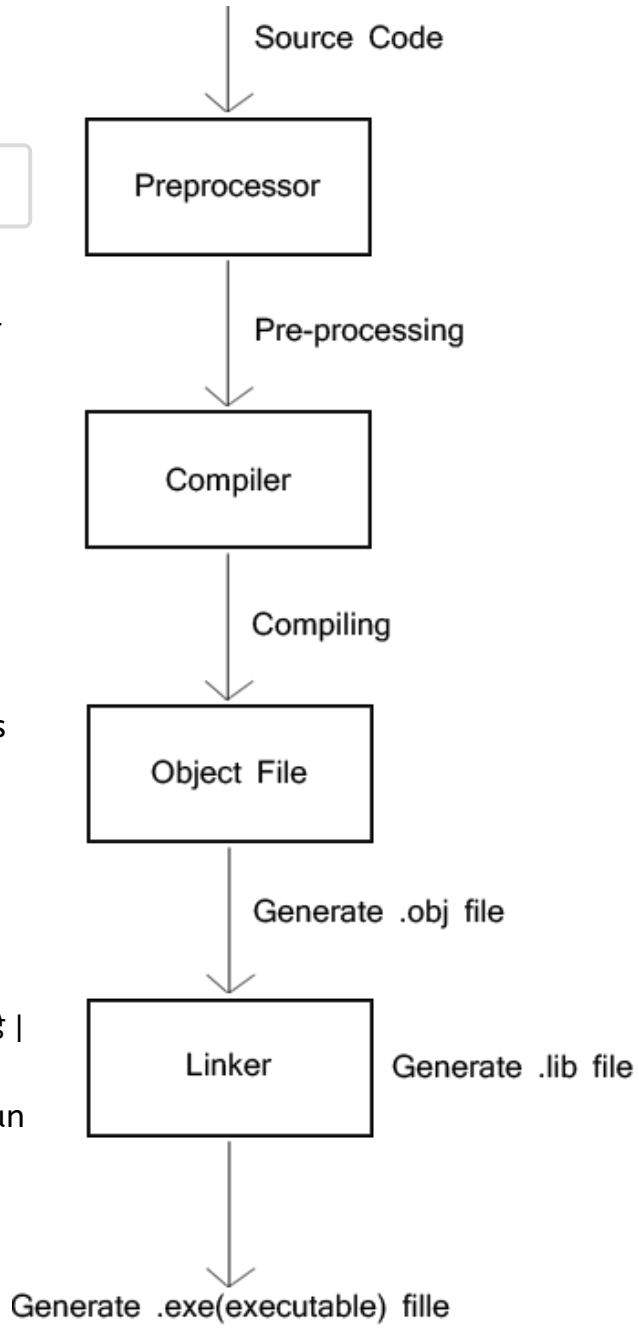
```
Enter Two Values
5
6
sum of 5 and 6 : 11
```

Command Line Argument

C Program को run करने के लिए Programmer अलग-अलग Compilers का इस्तेमाल करते है जैसेकि, Turbo C/C++, CodeBlock, Cfree और Dev C/C++ इत्यादी .

Compiler से program run कैसे होता है ?

Programmer पहले Source Code बनाता है, जिसका नाम Programmer कुछ भी रख सकता है, लेकिन .c extension/type लगाना अनिवार्य होता है | बाद में Preprocessor pre-process होता है | Pre-process होने के बाद Compiler के द्वारा Program को compile किया जाता है, Compile होने से जो IDE Programmer इस्तेमाल कर रहा है उसके द्वारा जिस file का नाम .c file type के साथ रखा है , उस file के नाम के साथ .obj extension की Object File तैयार होती है | बाद में Linker के साथ .lib library file तैयार होती है जो object file के साथ merge होती है | उसके बाद .exe executable file बनती है | जो हर Computer User अपने Computer में इस्तेमाल करता है |



Syntax for Command Line Arguments

```
int main( int argc, char *argv[])
```

Command Line के लिए main function में दो arguments होते है |

1. argc
2. argv[]

1. argc(Argument Counter)

ये integer type का variable या argument है |
इसमे arguments के total numbers main function को pass किये जाते है |
Command line में file के नाम को भी वो argument लेता है

2. argv[](Argument Vector)

ये character pointer type का argument होता है |
ये actual arguments को main function को pass किये जाते है |
ये सभी argument array बनाकर लेता है |
Command Line Arguments किसी भी IDE पर Compile या Run नहीं होता |
Command Line Arguments सिर्फ Command Prompt पर Run होते है |

Example for Command Line Argument

command.c

Source Code :

```
#include <stdio.h>
int main(int argc, char *argv[]){
int i;
printf("Number of Argument %d\n", argc);
for(i=1; i<argc; i++){
   printf("Argument %d = %s\n", i, argv[i]);
}
}
```

Command Line से Program Run कैसे किया जाता है ?

पहले Command Prompt को open करे |
Command Prompt को Open करने के लिए Window+R Press करे |
बाद में एक Run का Dialog Box खुल जायेगा उसमे 'cmd' नाम का Program डाले |Command Prompt खुल जाएगा |
Drive को change करने के लिए जिस Drive पर जाना है उस Drive का नाम और उसके साथ 'colon' दे | for eg.E:, C:, D:

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\UD>E:
E:\>
```

Directory को change करने के लिए 'cd' type करे और जिस directory पर जाना है उसका path दे |

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\UD>E:
E:\>cd E:\UD\Documents
E:\UD\Documents>
```

command.c

Source Code :

```
#include <stdio.h>
int main(int argc, char *argv[]){
int i;
printf("Number of Argument %d\n", argc);
for(i=1; i<argc; i++){
   printf("Argument %d = %s\n", i, argv[i]);
}
}
```

Program Command line से run करने से पहले IDE से run करे, अगर बिना error से program चले तो फिर Command Line से चलाये |

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\UD>E:
E:\>cd E:\UD\Documents
E:\UD\Documents>command Hi! Hello Friends
Number of Argument 4
Argument 1 : Hi!
Argument 2 : Hello
Argument 3 : Friends
E:\UD\Documents>
```

Example for Addition two Numbers using Command Line Arguments

commandline.c

Source Code :

```
#include <stdio.h>
int main(int argc, char * argv[]) {
int i, sum = 0;
if (argc != 3) {
    printf("Enter only Two Arguments.");
    exit (1);
}
    printf("Addition = %d", sum);
for (i = 1; i < argc; i++)
    sum = atoi(argv[1]) + atoi(argv[2]); // atoi is used to convert string to integer
    printf("%d", sum);
}
```

Output :

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\UD>E:
E:\>cd E:\UD\Documents
E:\UD\Documents>commandline 4 6
Addition = 010
E:\UD\Documents>
```

Header Files

stdio.h

Library Functions	Description
clearerr()	Error indicators को clear किया जाता है ।
fclose()	Open किये हुई file को close किया जाता है ।
feof()	End of file को ढूँढता है ।
fflush()	temporary data को clean कर देता है ।
fgetc()	File से character को read किया जाता है ।
fgetchar()	Keyboard से character read करता है ।
fgets()	File से string को read किया जाता है ।
fopen()	File को create और open किया जाता है ।
fprintf()	File में data को write किया जाता है ।
fputc()	File में character को write किया जाता है ।
fputchar()	Keyboard से character write करता है ।
fputs()	File में string को write किया जाता है ।
fscanf()	File में data को read किया जाता है ।
fseek()	दिए हुई position पर file pointer को set किया जाता है ।
ftell()	File Pointer की current position return करता है ।
getc()	Character को read करता है ।
getchar()	Keyboard से character read करता है ।
gets()	Keyboard से line को read करता है ।
getw()	File से integer को read किया जाता है ।
printf()	Data को screen पर write करने के लिए इस्तेमाल किया जाता है ।
putc()	Character को write किया जाता है ।

putchar()	Keyboard से character write करता है	
puts()	Keyboard से line को write करता है	
putw()	File में integer को write किया जाता है	
remove()	File को remove कर देता है	
rewind()	File Pointer को file के शुरुआत में लाता है	
scanf()	Numeric, character और string कोई भी value input किया जाता है	
sprintf()	Formatted Output String पर write किया जाता है	
sscanf()	Formatted input String पर read किया जाता है	

clearerr() : Standard Function

clearerr() इस function से end of file और error indication flags को clear किया जाता है |
clearerr() function कुछ भी return नहीं करता |
File जब open होता है तब indicators cleared हो जाते है |

Syntax

```
clearerr(file_pointer);
```

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
char str;
fptr = fopen("sample.txt", "w");
fgetc(fptr);
if(ferror(fptr)){
    printf("Reading Error from file\n");
}
clearerr(fptr);
if(ferror(fptr)){
      printf("Reading Error from file");
}
fclose(fptr);
return 0;
}
```

Output :

```
Reading Error from file
```

fclose() : Standard Function

fclose() funtion से open किये हुए file को close किया जाता है |

Syntax

```
fclose(file_pointer);
```

file_pointer : जिस file को close करना है उस file का pointer |

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
fptr = fopen("sample.txt","w");
fclose(fptr);
return 0;
}
```

feof() : Standard Function

feof() funtion से file end हुई है या नहीं हुई है ये check किया जाता है |

Syntax

```
feof(file_pointer);
```

Source Code :

```
#include <stdio.h>
int main(){
char str[20];
FILE *fptr;
fptr = fopen("sample.txt","r");
fgets(str, 20, fptr);
    printf("File Contents : %s\n", str);
if(feof(fptr)!=0){
    printf("File is ended.");
}
fclose(fptr);
return 0;
}
```

Output :

```
File Contents : Hello World!
File is ended.
```

fflush() : Standard Function

fflush() funtion से input या output buffer memory को clean कर देता है |
fflush() function ख़ास करके file के लिए ही इस्तेमाल किया जाता है |
File में जो data दिखाई नहीं देता या unwritten data होता है, वो fflush से clean कर दिया जाता है |

Syntax

```
fflush(file_pointer);
```

Source Code :

```
#include <stdio.h>
int main(){
int a, b, c;
    printf("Enter Value of a : ");
    scanf("%d", &a);
    printf("Enter Value of b : ");
fflush(stdin);
    scanf("%d", &b);
    printf("Value of a : %d\n", a);
    printf("Value of b : %d", b);
return 0;
}
```

Output :

```
Enter Value of a : 4 5 6    // 4 and 5 is cleaned
Enter Value of b : 5 6 7    // 6 and 7 is cleaned
Value of a : 4
Value of b : 5
```

fgetc() : Standard Function

fgetc() funtion से File में data को read किया जाता है |

Syntax

```
fgetc(file_pointer);
```

ये normally file में से single character ही return करता है | पर feof(end of file) होने के कारण जबतक file में से characters ख़त्म नहीं होते तबतक ये one-by-one characters print करते रहता है |

Source Code :

```
#include <stdio.h>
int main(){
char ch;
FILE *fptr;
fptr = fopen("sample.txt","r");
printf("File Contents : ");
while((!feof(fptr))){

ch = fgetc(fptr);
    printf("%c",ch);
}
fclose(fptr);
return 0;
}
```

Output :

```
File Contents : Hello World!
```

fgetchar() : Standard Function

fgetchar() funtion से एक character को read किया जाता है |
ये एक character keyboard से input लेता है |

Syntax

```
fgetchar(void);
```

Source Code :

```
#include <stdio.h>
int main(){
char c;
    printf("Enter Any character : ");
c = fgetchar();
    printf("Entered character is: ");
putchar(c);
return 0;
}
```

Output :

```
Enter Any character : h
Entered character is: h
```

fgets() : Standard Function

fgets ये File Handling का एक Function है | ये Function stdio.h इस header file के अंतर्गत आता है | यहाँ पर fgets का मतलब है File के string को read करना |

Syntax

```
fgets(char_array, size_of_char, file_pointer) ;
```

char_array - ये एक user-defined char data type का variable है | ये read करनेवाले string को अपने अंदर store करता है |
size_of_char - File से कितने character लेने है उनकी संख्या यहाँ पर लिखी जाती है |
file_pointer - File Pointer file को open/close करने के लिए use किया जाता है | File pointer के variable में fopen function लिखा जाता है |

sample.txt

```
Hello World!
```

Source Code :

```
#include <stdio.h>
int main(){
char str[20];
FILE *fptr;
fptr = fopen("sample.txt", "r");
    fgets(str, 20, fptr);
    printf("File Content : %s", str);
return 0;
}
```

Output :

```
File Contants : Hello World!
```

fopen() : Standard Function

fopen() funtion से File को create और open किया जाता है |

Syntax

```
file_pointer = fopen("file_name", "mode_of_file");
```

file_pointer : - ये एक file pointer है जिसका data type 'FILE' है |
file_name - File का नाम उसके extension के साथ देना चाहिए |for e.g. file.txt, file.docx .txt और .docx ये files के extensions है |
mode_of_file - File को किस mode से open करना ये उनका mode निश्चित करता है | File को Open करने के तीन basic modes है |

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
fptr = fopen("sample.txt","w");
return 0;
}
```

fprintf() : Standard Function

fprintf() funtion से File में data को write किया जाता है |

Syntax

```
fprintf(file_pointer, "format_specifier or string", variables);
```

file_pointer : - जिस file को write करना है उस file का pointer |
"format_specifier or string" - जैसे printf में इस्तेमाल किया जाता है, वैसे ही यहाँ format specifier और string का इस्तेमाल यहाँ पर होता है | दोनों एक साथ भी इस्तेमाल किये जा सकते है |
variables - अगर format specifier जहाँ पर दिया जाता है, वह पर variable तो होते ही है | अगर format specifier नहीं होते तो variables भी नहीं होते |

Source Code :

```
#include <stdio.h>
int main(){
int emp_id;
char emp_name[20];
FILE *fptr;
fptr = fopen( "sample.txt", "w" );
printf("Enter Employee ID :");
scanf("%d", &emp_id);
printf("Enter Employee Name :");
scanf("%s", &emp_name);
fprintf( fptr, "%d, %s", emp_id, emp_name ) ;
fclose(fptr);
return 0;
}
```

Output :

```
Enter Employee ID :2
Enter Employee Name :Raj
```

sample.txt

```
2, Raj
```

fputc() : Standard Function

fputc() funtion से File में data को एक character को write किया जाता है |

Syntax

```
fputc(char c, file_pointer);
```

char c : - ये वो character है जिसे file में write किया जाता है |
file_pointer - ये एक file pointer है जिसमे उस character को write करना है |

Source Code :

```
#include <stdio.h>
int main(){
char c;
    printf("Enter Any character : ");
c = fgetchar();
    printf("Entered character is: ");
putchar(c);
return 0;
}
```

Output :

```
Enter Any character : h
Entered character is: h
```

fputchar() : Standard Function

fputchar() funtion से एक character को write किया जाता है |

Syntax

```
fputchar(character);
```

Source Code :

```
#include <stdio.h>
int main(){
char c;
for(c=32; c<127; c++){
    fputchar(c);
}
return 0;
}
```

Output :

```
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

fputs() : Standard Function

fputs ये File Handling का एक Function है | ये Function stdio.h इस header file के अंतर्गत आता है | यहाँ पर fputs का मतलब है File में string को write करना |

Syntax

```
fputs(char_array, file_pointer) ;
```

char_array - ये एक user-defined char data type का variable है | यहाँ पर जो string लिखा जाता वो fopen में लिखे हुए file में store हो जाता है |
file_pointer - जिस file पर write करना है उस file का pointer यहाँ आता है |

Sample.txt

```
Hello World!
```

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
char str[20] = "Hello World";
fptr = fopen("sample.txt", "w");
fputs(str, fptr);
fclose(fptr);
return 0;
}
```

Output :
Sample.txt

```
Hello World!
```

fscanf() : Standard Function

fscanf() funtion से File में data को read किया जाता है |

Syntax

```
fscanf(file_pointer, "format_specifier or string", &variables);
```

file_pointer : - जिस file को read करना है उस file का pointer |
"format_specifier or string" - जैसे scanf में इस्तेमाल किया जाता है, वैसे ही यहाँ format specifier और string का इस्तेमाल यहाँ पर होता है | दोनों एक साथ भी इस्तेमाल किये जा सकते है |
&variables - अगर format specifier जहाँ पर दिया जाता है, वह पर variable तो होते ही है | अगर format specifier नहीं होते तो variables भी नहीं होते |

sample.txt

```
2, Raj
```

Source Code :

```
#include <stdio.h>
int main(){
int emp_id;
char emp_name[20];
FILE *fptr;
fptr = fopen( "sample.txt", "r" );
fscanf( fptr, "%d,%s", &emp_id, emp_name ) ;
    printf("Employee ID : %d\n", emp_id);
    printf("Employee Name : %s", emp_name);
fclose(fptr);
return 0;
}
```

Output :

```
Employee ID : 2
Employee Name : Raj
```

fseek() : Standard Function

fseek() funtion से दिए हुई position पर file pointer को set किया जाता है |

Syntax

```
fseek(file_pointer, offset, position);
```

file_pointer : - जिस file को read और write करना है उस file का pointer |
offset - File के data को beginning(SEEK_SET) से, end(SEEK_END) से और current(SEEK_CUR) position से कितने characters या bytes replace या store करना है वो integer value यहाँ पर आती है |
position - File के data को कहा से replace करे या store करे ये वो position है |for eg. SEEK_SET(0), SEEK_CUR(1), SEEK_END(2)

sample.txt

```
Hello World
```

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
char str[20];
fptr = fopen("sample.txt","r+");
fgets(str, 20, fptr);
   printf("Old file contents : %s\n", str);   // Old contents : Hello World
fseek(fptr, 6, SEEK_SET);   // replace contents after 6 characters
fputs("Friends", fptr);     // put Friends after 6 characters
fclose(fptr);    //file is closed
fptr = fopen("sample.txt","r");   // file is reopened
                                  // User can use rewind() function instead of reopening file
fgets(str, 20, fptr);
   printf("New file contents : %s\n", str);
fclose(fptr);
return 0;
}
```

Output :

```
Old file contents : Hello World
New file contents : Hello Friends
```

sample.txt

```
Hello Friends
```

ftell() : Standard Function

ftell() funtion File Pointer की current position return करता है |

Syntax

```
ftell(file_pointer);
```

file_pointer : - ये एक file pointer है | जिससे file की current position समझ आएगी |

ये long integer को return करता है | इससे file के data को bytes में नापता है | अगर कुछ कारन से file open होने में दिक्कत आती है, तो ये -1 return करता है |

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
char str[] = "Hello World";
fptr=fopen("file.txt","w");
fputs(str,fptr);
    printf("File Contents : %s\n",str);
int size = ftell(fptr);
    printf("Size of file in bytes : %d\n", size);
fclose(fptr);
return 0;
}
```

Output :

```
File Contents : Hello World
Size of file in bytes : 11
```

getc() : Standard Function

getc() funtion से File में character को read किया जाता है |

Syntax

```
getc(file_pointer);
```

sample.txt

```
Hello World!
```

Source Code :

```
#include <stdio.h>
int main(){
char ch;
FILE *fptr;
fptr = fopen("sample.txt","r");
while((ch=getc(fptr))!=EOF){
    putchar(ch);
}
fclose(fptr);
return 0;
}
```

Output :

```
Hello World!
```

getchar() : Standard Function

getchar() funtion से एक character को input किया जाता है |
ये एक character keyboard से input लेता है |

Syntax

```
getchar(void);
```

Source Code :

```
#include <stdio.h>
int main(){
char c;
printf("Enter a character :");
c = getchar();
printf( "You entered character is : ");
putchar(c);
return 0;
}
```

Output :

```
Enter a character :h
You entered character is : h
```

gets() : Standard Function

gets का इस्तेमाल user से input मतलब string लेने की अनुमति ली जाती है | puts input लिए string को output में print करता है |

Syntax

```
gets(string);
```

Source Code :

```
#include <stdio.h>
int main()
{
char name[20];
    printf( "Enter your name : ");
gets(name);
    printf( "You name is : ");
puts(name);
return 0;
}
```

Output :

```
Enter your name : Raj
You name is : Raj
```

getw() : Standard Function

getw() function से file में integer value को read किया जाता है |

Syntax

```
getw(file_pointer);
```

sample.txt

```
1
```

Source Code :

```
#include <stdio.h>
int main(){
int i;
FILE *fptr;
fptr=fopen("sample.txt","r");
while((i=getw(fptr)!=EOF))
    printf("%d\n",i);
fclose(fptr);
return 0;
}
```

Output :

```
1
```

printf() : Standard Function

printf() function से data को screen पर write करने के लिए इस्तेमाल किया जाता है |

Syntax

```
printf("String / format_specifier(s) / escape sequence(s)", variable_list(s) );
```

Source Code :

```
#include <stdio.h>
int main(){
int i = 5;
    printf("Value of i is %d\", i);
return 0;
}
```

Output :

```
Value of i is 5
```

putc() : Standard Function

putc() funtion से File में character को write किया जाता है |

Syntax

```
putc(character, file_pointer);
```

Source Code :

```
#include <stdio.h>
int main ()
{
FILE *fptr;
char c = 'H';
fptr = fopen("sample.txt", "w");
putc(c, fptr);
fclose(fptr);
return 0;
}
```

sample.txt

```
H
```

putchar() : Standard Function

putchar() funtion से input किये हुए character को output में print करता है |

Syntax

```
putchar(character);
```

Source Code :

```
#include <stdio.h>
int main(){
char c;
    printf( "Enter a character :");
c = getchar();
    printf( "You entered character is : ");
putchar(c);
return 0;
}
```

Output :

```
Enter a character :h
You entered character is : h
```

puts() : Standard Function

puts का इस्तेमाल user से input मतलब string लेने की अनुमति ली जाती है | puts input लिए string को output में print करता है |

Syntax

```
puts(string);
```

Source Code :

```
#include <stdio.h>
int main()
{
char name[20];
    printf( "Enter your name : ");
puts(name);
    printf( "You name is : ");
puts(name);
return 0;
}
```

Output :

```
Enter your name : Raj
You name is : Raj
```

putw() : Standard Function

putw() function से file में integer value को write किया जाता है |

Syntax

```
putw(integer, file_pointer);
```

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
int i = 1;
fptr = fopen("sample.txt","w");
putw(i, fptr);
fclose(fptr);
return 0;
}
```

sample.txt

```
1
```

remove() : Standard Function

ove() funtion से file को delete या remove किया जाता है |

Syntax

```
remove(file_name);
```

file_name : जिस file को remove करना उस file का नाम यहाँ पर आएगा |

जिस directory पर programmer निचे दिया हुआ source code save करता है, उस directory की file ही delete की जाती है | अगर file को delete करना हो तो उस file को उसके extension/type के साथ देना पड़ता है | file को delete करते वक्त सोचे कि ये file permanantly delete हो जायेगी | यहाँ से delete होने वाली file 'Recycle Bin' में नहीं जाती |

Source Code :

```
#include <stdio.h>
int main(){
int remover;
char file[20];
    printf("Enter file name to delete : ");
    scanf("%s", file);
remover = remove(file);
if(remover == 0){
    printf("File is removed.");
}else{
    printf("Unable to delete file.");
}
return 0;
}
```

output :

```
Enter file name to delete : hello.c
Unable to delete file.
```

rewind() : Standard Function

rewind() funtion से file pointer को file के शुरुआत में लाता है |

Syntax

```
rewind(file_pointer);
```

file pointer : जिस file को इस file pointer में open किया है, उस file pointer को शुरुआत में लाता है |

sample.txt

```
Hello World!
```

Source Code :

```
#include <stdio.h>
int main(){
FILE *fptr;
char str[20];
fptr = fopen("file.txt","r+");
fgets(str, 20, fptr);
    printf("Old file contents : %s\n", str);
fseek(fptr, 6, SEEK_SET);
fputs("Friends", fptr);
rewind(fptr);   // file is reopened
fgets(str, 20, fptr);
    printf("New file contents : %s\n", str);
fclose(fptr);
return 0;
}
```

Output :

```
Old File Contents : Hello World
New file contents : Hello Friends
```

scanf() : Standard Function

scanf() function के जरिये numeric, characters और string कोई भी value input किया जाता है |

Syntax

```
scanf("format_specifier(s)", &variable_list(s) );
```

Source Code :

```
#include <stdio.h>
int main(){
int a;
    printf("Enter value of a : ");
    scanf("%d", &a);
    printf("Value of a is %d", a);
return 0;
}
```

Output :

```
Enter value of a : 5
Value of a is 5
```

sprintf() : Standard Function

Syntax

```
sprintf(string, format_specifier(s), variables);
```

string : इस string पर जो buffer है वो store किया जाता है |
format_specifier(s) : data type के हिसाब से यहाँ पर format specifier दिया जाता है |
variables : जब format specifiers का इस्तेमाल होता है, तब इन variables का इस्तेमाल होता है |

Source Code :

```
#include <stdio.h>
int main()
{
char str1[20] = "Hello World!";
char str2[20] = "Hello Friends!";
  printf("String before sprintf : %s\n", str1);
  sprintf(str1, "%s", str2);
  printf("String after sprintf : %s\n", str1);
  return 0;
}
```

Output :

```
String before sprintf : Hello World!
String after sprintf : Hello Friends!
```

sscanf() : Standard Function

Syntax

```
sscanf(string, format_specifier(s), &variables);
```

Source Code :

```
#include <stdio.h>
int main ()
{
char mainstr[50]="I am 21 years old";
char str1[10];
char str2[10];
int i;
char str3[10];
char str4[10];
   sscanf(mainstr, "%s %s %d %s %s", str1, str2, &i, str3, str4);
   printf("str1 = %s\n", str1);
   printf("str2 = %s\n", str2);
   printf("i = %d\n", i);
   printf("str3 = %s\n", str3);
   printf("str4 = %s", str4);
return 0;
}
```

Output :

```
str1 = I
str2 = am
i = 21
str3 = years
str4 = old
```

conio.h

Library Functions	Description
clrscr()	Output Screen को clear किया जाता है ।
delline()	Program के line को delete किया जाता है ।
getch()	Keyboard से character को read किया जाता है।
getche()	Output screen के लिए Keyboard से एक character लेने की अनुमति ली जाती है ।
gotoxy()	gotoxy() function cursor को move कर देता है ।
kbhit()	Keyboard का button press हुआ है या नहीं हुआ है ये निर्धारित करता है ।
textbackground()	Text का background color change करने के लिए इस्तेमाल होता है ।
textcolor()	Text का color change करने के लिए इस्तेमाल होता है ।
wherex()	Horizontal में current position integer में return करता है ।
wherey()	Vertical में current position integer में return करता है ।

clrscr() : Console Library Function

clrscr() function से output screen को clear किया जाता है ।

Syntax

```
clrscr();
```

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
printf("Hello World\n");
clrscr();
printf("Hello Friends");
getch();
return 0;
}
```

Output :

```
Hello Friends
```

delline() : Console Library Function

delline() function से लाइन को डिलीट किया जाता है |

Syntax

```
delline()
```

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
clrscr();
printf("Hello World");
getch();
delline();
printf("Hello World line is deleted.");
getch();
return 0;
}
```

Output :

```
Hello World line is deleted.
```

getch() : Console Library Function

getch() function User से एक character आने का wait करता है |
लेकिन वो output में print नहीं होता |getch() function कुछ भी return नहीं करता |

Syntax

```
int getch(void);
```

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
    printf("Enter any character to exit.\n");
getch();
return 0;
}
```

Output :

```
Enter any character to exit.
```

getche() : Console Library Function

getche() function User से एक character आने का wait करता है | लेकिन वो output में print होता है |

Syntax

```
int getche(void);
```

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
    printf("Enter any character to exit.\n");
getche();
return 0;
}
```

Output :

```
Enter any character to exit.
g
```

gotoxy() : Console Library Function

gotoxy() function cursor को move कर देता है |

Syntax

```
gotoxy(int x, int y);
```

gotoxy() function के दो parameters होते है |

x (x-coordinate) - ये integer value होता है | ये Horizontal position मतलब cursor को left से right लेने के इसका इस्तेमाल होता है |

y (y-coordinate)- ये integer value होता है | ये Vertical position मतलब cursor को top से bottom लेने के इसका इस्तेमाल होता है |

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
gotoxy(10, 10);
    printf("cursor position is changed.");
getch();
return 0;
}
```

Output :

kbhit() : Console Library Function

kbhit() function keyboard का button hit किया है या नहीं ये निर्धारित करता है |

Syntax

```
kbhit();
```

जब Keyboard की key press हो जाती है, तब non-zero value return होती है नहीं तो zero return करता है |

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
while(!kbhit())
    printf("You haven't pressed a key.\n");
getch();
return 0;
}
```

Output :

```
You haven't pressed a key.
You haven't pressed a key.
You haven't pressed a key.
You haven't pressed a key.
You haven't pressed a key.
You haven't pressed a key.
You haven't pressed a key.
You haven't pressed a key.
You haven't pressed a key.
You haven't pressed a key.
```

textbackground() : Console Library Function

textbackground() function से text के background को color किया जाता है |
color का नाम Uppercase में लिखा जाता है |

Syntax

```
textbackground(color_name or value);
```

Color Name	Color Value
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	9
LIGHTCYAN	10
LIGHTRED	11
LIGHTMAGENTA	12
YELLOW	13
WHITE	14

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
textbackground(BLUE);
cprintf("Hello World!");    // cprintf specially use for console(screen)
getch();
return 0;
}
```

Output :

```
Hello World!
```

textcolor() : Console Library Function

textcolor() function से text को color किया जाता है |
color का नाम Uppercase में लिखा जाता है |

Syntax

```
textcolor(color_name or value);
```

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
textcolor(BLUE);
cprintf("Hello World!\n");    // cprintf specially use for console(screen)
textcolor(RED+128);     // 128 is value of BLINK
cprintf("Blinking Text");
getch();
return 0;
}
```

Output :

```
Hello World!
Blinking Text
```

wherex() : Console Library Function

wherex() function Horizontal में cursor की current position को return करता है |

Syntax

```
int wherex();
```

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
int x ;
clrscr();
x = wherex();
printf("Current Position of cursor is %d.", x);
getch();
return 0;
}
```

Output :

```
Current Position of cursor is 1.
```

wherey() : Console Library Function

wherey() function Vertical में cursor की current position को return करता है |

Syntax

```
int wherey();
```

Source Code :

```
#include <stdio.h>
#include <conio.h>
int main(){
int x ;
clrscr();
x = wherey();
printf("Current Position of cursor is %d.", x);
getch();
return 0;
}
```

Output :

```
Current Position of cursor is 1.
```

stdlib.h

Library Functions	Description
abort()	C Program को terminate करने का काम करता है ।
abs()	Absolute value को return किया जाता है ।
atof()	String data type को float data type में convert किया जाता है ।
atoi()	String data type को integer data type में convert किया जाता है ।
atol()	String data type को long data type में convert किया जाता है ।
calloc()	एक से अधिक Memory Blocks बनाने के लिए काम आता है ।
div()	दिए हुए numerator और denominator से remainder और quotient को निकला जाता है ।
exit()	Program को terminate करने का काम करता है ।
free()	malloc() और calloc() से जो memory allocate की गयी है, उसे deallocate कर देता है ।
malloc()	एक ही Memory Block बनाने के काम आता है ।
perror()	एक error handing function है ।
rand()	Random Number को 0 से RAND_MAX तक दिखता है ।
realloc()	अगर malloc() और calloc() की memory को बढाने और घटाने का काम करता है ।
strtod()	String data type को double data type में convert किया जाता है ।
strtol()	String data type को long data type में convert किया जाता है ।
system()	Operating system के command के लिए काम आता है ।

abort() : Standard Library Function

abort() function से Program के execution को terminate कर देता है |
abort() function कुछ भी return नही करता |

Syntax

```
abort(void);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
FILE * fptr;
fptr= fopen ("sample1.txt","r");
if (fptr == NULL){
    printf("File is not opened.");
abort();
}
fclose (fptr);
return 0;
}
```

Output :

```
File is not opened.
```

abs() : Standard Library Function

abs() Absolute value को return करता है |

Syntax

```
abs(Number);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
int a ;
a = 5;
    printf("Absolute value of %d is %d\n", a, abs(5));
a = -5;
    printf("Absolute value of %d is %d\n", a, abs(5));
return 0;
}
```

Output :

```
Absolute value of 5 is 5
Absolute value of -5 is 5
```

atof() : Standard Library Function

atof() इस function से character data type को float-point(double) data type में convert किया जाता है |

Syntax

```
atof(string);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
char str[10] = "0.01";
float i;
 i = atof(str);
    printf("i = %f\n", i);
return 0;
}
```

Output :

```
i = 0.010000
```

atoi() : Standard Library Function

atoi() इस function से character data type को int data type में convert किया जाता है |

Syntax

```
atoi(string);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
char str[10] = "0.01";
int i;
 i = atoi(str);
    printf("i = %d\n", i);
return 0;
}
```

Output :

```
i = 0
```

atol() : Standard Library Function

atol() इस function से character data type को long data type में convert किया जाता है |

Syntax

```
long atol(string);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
char str[10] = "110000000000";
long i;
 i = atol(str);
    printf("i = %ld\n", i);
return 0;
}
```

Output :

```
i = 1100000000
```

calloc()(Contiguous Allocation): Standard Library Function

calloc() ये Memory का block array में allocate करता है |

Syntax

```
pointer = (cast_type*)calloc(number_of_block, size_of_per_block);
```

calloc() ये function एक Dynamic Memory Allocation का प्रकार है |
calloc() function में multiple Memory blocks बनाता है इसीलिए इसे दो parameters दिए हुए है |
calloc() function void pointer return करता है |
calloc() function में जब Memory allocate होती है, तब इसकी हर block की initial value '0' होती है |
जब इस pointer को memory ये space नहीं मिलती, तो वो NULL return करता है |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int a, i;
int *ptr;
    printf("Enter Number of Elements : ");
    scanf("%d",&a);
ptr = (int*)calloc(a,sizeof(int));      // calloc memory allocating
if(ptr==NULL){
    printf("Error memory allocating");
    exit (0);
}
for(i=0; i<a; i++){
    printf("Enter Element Number %d : ", i+1);
scanf("%d",&ptr[i]);
}
for(i=0; i<a; i++){
printf("Element %d : %d\n", i+1, ptr[i]);
}
free(ptr);
return 0;
}
```

Output :

```
Enter Number of Elements : 4
Enter Element Number 1 : 5
Enter Element Number 2 : 6
Enter Element Number 3 : 8
Enter Element Number 4 : 7
Element 1 : 5
Element 2 : 6
Element 3 : 8
Element 4 : 7
```

Memory Allocation Storage

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int a, i;
int *ptr;
    printf("Enter Number of Elements : ");
    scanf("%d",&a);
ptr = (int*)calloc(a,sizeof(int));
if(ptr==NULL){
    printf("Error memory allocating");
    exit (0);
}
return 0;
}
```

Output :

```
Enter Number of Elements : 13113331131444
Error memory allocating
```

div(): Standard Library Function

div() ये function quotient और remainder return करता है |

Syntax

```
div_t div(numerator, denominator)
```

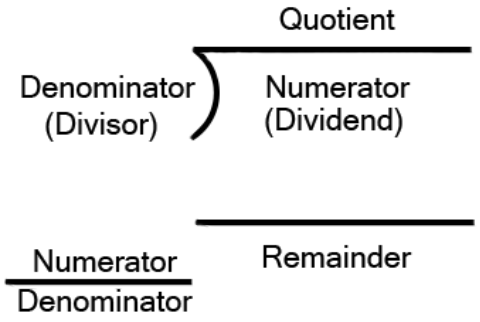
Remainder के लिए div_t.rem का इस्तेमाल किया जाता है |
Quotient के लिए div_t.quot का इस्तेमाल किया जाता है |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
div_t num;
num = div(5,2);
   printf("Quotient of 5/2 is %d\n", num.quot);
   printf("Remainder of 5/2 is %d", num.rem);
return 0;
}
```

Output :

```
Quotient of 5/2 is 2
Remainder of 5/2 is 1
```



exit(): Standard Library Function

exit() ये function जब call होता है, तब Program को terminate कर देता है |

Syntax

```
exit(status);
```

'0'(EXIT_SUCCESS) इस status से Program बिना error के exit हो जाता है |
'1'(EXIT_FAILURE) इस status से Program के errors को handle किया जाता है |
exit() function कोई value return नहीं करता बल्कि उसका termination status return करता है |

exit() function क्या-क्या करता है ?
File के लिए, अगर file को open किया गया है, तो close किया जाता है |
Temporary data को clean कर देता है |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int i;
FILE *fptr;
if ((fptr = fopen("sample.txt","r")) == NULL){
    printf("Error! opening file");
    exit(1);
}
    printf("Garbage value of i is %d", i);
fclose(fptr);
return 0;
}
```

Output :

```
Error! opening file
Process returned 1 (0x1)   execution time : 0.047 s
```

free(): Standard Library Function

free() function से malloc, calloc और realloc से जो Memory allocate की गयी है , उसे deallocate या free कर देता है |

Syntax

```
free(name_of_pointer);
```

अगर Pointer NULL होता है, तो free() function कार्यान्वित नहीं होता |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int a, i;
int *ptr;
    printf("Enter Number of Elements : ");
    scanf("%d",&a);
ptr = (int*)calloc(a,sizeof(int));      // calloc memory allocating
if(ptr==NULL){
    printf("Error memory allocating");
    exit (0);
}
for(i=0; i<a; i++){
    printf("Enter Element Number %d : ", i+1);
scanf("%d",&ptr[i]);
}
for(i=0; i<a; i++){
printf("Element %d : %d\n", i+1, ptr[i]);
}
free(ptr);
return 0;
}
```

Output :

```
Error! opening file
Process returned 1 (0x1)   execution time : 0.047 s
```

malloc() (Memory Allocation): Standard Library Function

malloc() ये Memory का block allocate करता है |

Syntax

```
pointer = (cast_type*)malloc(size_of_block);
```

calloc() ये function एक Dynamic Memory Allocation का प्रकार है |
malloc() function सिर्फ एक Memory block allocate करता है |
अगर ये memory allocate नहीं करता तो वो NULL return करता है |
malloc() function में जब Memory allocate होती है, तब इसकी block पर garbage value होती है |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int a, i;
int *ptr;
    printf("Enter Number of Elements : ");
    scanf("%d",&a);
ptr = (int*)malloc(a*sizeof(int));
if(ptr==NULL){
    printf("Error memory allocating");
    exit (0);
}
for(i=0; i<a; i++){
    printf("Enter Element Number %d : ", i+1);
scanf("%d",&ptr[i]);
}
for(i=0; i<a; i++){
printf("Element %d : %d\n", i+1, ptr[i]);
}
free(ptr);
return 0;
}
```

Output :

```
Enter Number of Elements : 4
Enter Element Number 1 : 5
Enter Element Number 2 : 6
Enter Element Number 3 : 8
Enter Element Number 4 : 7
Element 1 : 5
Element 2 : 6
Element 3 : 8
Element 4 : 7
```

Memory Allocation Storage

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int a, i;
int *ptr;
    printf("Enter Number of Elements : ");
    scanf("%d",&a);
ptr = (int*)malloc(a,sizeof(int));
if(ptr==NULL){
    printf("Error memory allocating");
    exit (0);
}
return 0;
}
```

Output :

```
Enter Number of Elements : 13113331131444
Error memory allocating
```

perror(): Standard Library Function

perror() ये function एक Error Handling का प्रकार है |

Syntax

```
void perror(const char *string);
```

ये function Output में एक error का message display करता है | इसके लिए उसे सिर्फ double quotes("") की जरुरत होती है |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
FILE *fptr;
fptr = fopen("nofile.txt", "r");
if(fptr == NULL){
printf("Error found\n");
    perror("");
}
fclose(fptr);
return 0;
}
```

Output :

```
Error found
No such file or directory
```

rand(): Standard Library Function

rand() ये function एक Error Handling का प्रकार है |

Syntax

```
rand()
```

rand() function random value 0 to RAND_MAX तक दिखाता है |
ये RAND_MAX stdlib.h इस header file में define किया गया है |
#define RAND_MAX 0x7FFFRAND
_MAX को hexadecimal macro दिया गया है | उस Hexadecimal को decimal में convert कर दिया तो 32767 ये decimal value मिलती है | इसका मतलब rand() function 0 से 32767 तक random number दिखाता है |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int i, n;
   printf("Random Numbers between 1 to 100\n");
for (i = 0; i < 10; i++) {
n = rand() % 100+1;
   printf("%d\n", n);
}
return 0;
}
```

Output :

```
Random Numbers between 1 to 100
42
68
35
1
70
25
79
59
63
65
```

realloc(): Standard Library Function

realloc() ये function एक Dynamic Memory Allocation का प्रकार है |
realloc() function void pointer return करता है |
realloc() ये function malloc() और calloc() function की memory को बढाया या घटाया जाता है |

Syntax

```
new_or_old_pointer = (cast_type*) realloc(pointer, new_size);
```

new_or_old_pointer : ये एक pointer है , जिससे realloc जो नयी memory allocate करेगा, उसके लिए ये pointer बनाया है |
pointer : जिस pointer की memory को reallocate करना हो तो उसका pointer यहाँ आएगा, चाहे वो calloc हो या malloc हो |
new_size : यहाँ पर calloc या malloc के लिए नयी size दी जाती है जो कम भी हो सकती है और ज्यादा भी हो सकती है |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int a, b, i;
int *ptr;
    printf("Enter Number of Elements : ");
    scanf("%d",&a);
ptr = (int*)malloc(a*sizeof(int));      // malloc memory allocating
if(ptr==NULL){
    printf("Error memory allocating");
    exit(0);
}
for(i=0; i<a; i++){
    printf("Enter Element Number %d : ", i+1);
scanf("%d",&ptr[i]);
}
for(i=0; i<a; i++){
    printf("Element %d : %d\n", i+1, ptr[i]);
}
    printf("\nEnter Number of New Elements : ");
    scanf("%d", &b);
ptr = (int*)realloc(ptr, b*sizeof(int));   // reallocating memory
if(ptr==NULL){
    printf("Error memory allocating");
    exit(0);
}
for(i=0; i<b; i++){
    printf("Enter Element Number %d : ", i+1);
scanf("%d",&ptr[i]);
}
```

```
for(i=0; i<b; i++){
    printf("Element %d : %d\n", i+1, ptr[i]);
}
free(ptr);
return 0;
}
```

Output :

```
Enter Number of Elements : 2
Enter Element Number 1 : 21
Enter Element Number 2 : 22
Element 1 : 21
Element 2 : 22
Enter Number of New Elements : 3
Enter Element Number 1 : 54
Enter Element Number 2 : 23
Enter Element Number 3 : 41
Element 1 : 54
Element 2 : 23
Element 3 : 41
```

strtod(): Standard Library Function

strtod() function string data type को double data type में convert कर देता है |

Syntax

```
double strtod(const char * str, char ** endptr);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
double num;
char str1[30] = "89 marks in Math";
char *ptr;
num = strtod(str1, &ptr);
    printf("String : %s\n" , ptr);
    printf("double value : %.2f\n", num);
return 0;
}
```

Output :

```
String : marks in Math
double value : 89.00
```

strtol(): Standard Library Function

strtol() function string data type को long data type में convert कर देता है |

Syntax

```
double strtol(const char * str, char ** endptr);
```

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
long num;
char str1[30] = "89 marks in Math";
char *ptr;
num = strtol(str1, &ptr, 10);
    printf("String : %s\n" , ptr);
    printf("long value : %ld\n", num);
return 0;
}
```

Output :

```
String :  marks in Math
double value : 89
```

system(): Standard Library Function

system() function Operating system पर काम करता है |

Syntax

```
int system(str command_pointer);
```

system() function में operting system के command का नाम होता है |

Source Code :

```
#include <stdio.h>
#include <stdlib.h>
int main(){
char command[20];
    printf("Enter command name : ");
    scanf("%s", command);
system(command);
return(0);
}
```

Output :

```
Enter command name : control
```

math.h

Library Functions	Description
acos()	Arc cosine radians में returns करता है	
acosh()	Hyperbolic arc cosine radians में return करता है	
asin()	Arc sine Number को radians में return करता है	
asinh()	Arc hyperbolic sine Number को radians में return करता है	
atan()	Arc tangent Number को radians में return करता है	
atanh()	Hyperbolic Arc tangent Number को radians में return करता है	
atan2()	Arc tangent Number को radians में return करता है	
cbrt()	Arc tangent2 Number को radians में return करता है	
ceil()	ceil() में दिया हुआ Number अपने पासवाले या उससे बड़े या उसके बराबर के integer Number को return करता है	
cos()	cos() function में cosine का angle radians में return करता है	
cosh()	Hyperbolic cosine का angle radians में return करता है	
exp()	e को घातांक दिया जाता है	ये double data type की value return करता है	
fabs()	fabs() ये function absolute value को return करता है	
floor()	floor() ये function अपने पासवाले या दिए हुए number से छोटे या इससे बराबर के integer number को return करता है	
hypot()	ये दो side के values लेकर hypotenuse की length निकालता है	
log()	log() ये function Number का natural logarithm return करता है	
log10()	log() ये function Number का base-10 logarithm return करता है	
pow()	pow() ये function 'a' ये value होती है और 'b' ये उस value का घातांक होता है	
sqrt()		
round()	round() ये function Number की round value का integer return करता है	
sin()	sin() ये function sine का angle radians में return करता है	
sqrt()	sqrt() ये function Number का square root return करता है	
tan()	tan() ये function Number का tangent का angle radians में return करता है	
tanh()	Arc hyperbolic tangent radians में return करता है	
trunc()	trunc() ये function Number को truncated integer Number में convert करता है	

acos() : Math Function

Arc cosine radians में return करता है |
Arc Cosine में जो Number दिया जाता है वो -1 to +1 के बीच वाला होना चाहिए |

Syntax

```
double acos(double Number)
```

Arc cosine ये radians को return करता है | वो radians 0 to PI(3.145) के बीच का होता है |
अगर radians से degree में convert करना हो तो 180/PI() से multiply करना पड़ता है |

Source Code :

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415
int main(){
double Number, r;
Number =  -0.5;
r = acos(Number);
   printf("Arc cosine of %.2lf = %.2lf in radians\n", Number, r);
r = acos(Number)*180/PI;   //radians to degree
   printf("Arc cosine of %.2lf = %.2lf in degree\n", Number, r);
return 0;
}
```

Output :

```
Arc cosine of -0.50 = 2.09 in radians
Arc cosine of -0.50 = 120.00 in degree
```

acosh() : Math Function

Arc hyperbolic cosine radians में return करता है |
Arc hyperbolic Cosine में जो Number दिया जाता है वो 1 या 1 से बड़ा होता है |

Syntax

```
double acosh(double Number)
```

अगर radians से degree में convert करना हो तो 180/PI() से multiply करना पड़ता है |

Source Code :

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415
int main(){
double Number, r;
Number =  3;
r = acosh(Number);
   printf("Arc hyperbolic cosine of %.2lf = %.2lf in radians\n", Number, r);
r = acosh(Number)*180/PI;   //radians to degree
   printf("Arc hyperbolic cosine of %.2lf = %.2lf in degree\n", Number, r);
r = acosh(0.5);
   printf("%.2lf = %.2lf in degree\n", Number, r);
return 0;
}
```

Output :

```
Arc hyperbolic cosine of 3.00 = 1.76 in radians
Arc hyperbolic cosine of 3.00 = 101.00 in degree
3.00 = 1.#R in degree
```

asin() : Math Function

Arc sine Number को radians में return करता है |
Arc sine में जो Number दिया जाता है वो -1 to +1 के बीच वाला होना चाहिए |

Syntax

```
double asin(double Number)
```

अगर radians से degree में convert करना हो तो 180/PI() से multiply करना पड़ता है |

Source Code :

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415
int main(){
double Number, r;
Number = 0.5;
r = asin(Number);
   printf("Arc sine of %.2lf = %.2lf in radians\n", Number, r);
r = asin(Number)*180/PI;   //radians to degree
   printf("Arc sine of %.2lf = %.2lf in degree\n", Number, r);
return 0;
}
```

Output :

```
Arc sine of 0.50 = 0.52 in radians
Arc sine of 0.50 = 30.00 in degree
```

asinh() : Math Function

Arc hyperbolic sine Number को radians में return करता है |
Arc hyperbolic sine में जो Number दिया जाता है वो कुछ भी हो सकता है |

Syntax

```
double asinh(double Number)
```

Source Code :

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415
int main(){
double Number, r;
Number =  5;
r = asinh(Number);
   printf("Arc hyperbolic cosine of %.2lf = %.2lf in radians\n", Number, r);
r = asinh(Number)*180/PI;   //radians to degree
   printf("Arc hyperbolic cosine of %.2lf = %.2lf in degree\n", Number, r);
return 0;
}
```

Output :

```
Arc hyperbolic cosine of 5.00 = 2.31 in radians
Arc hyperbolic cosine of 5.00 = 132.50 in degree
```

atan() : Math Function

Arc tangent negative या positive value radians में return करता है |

Syntax

```
double atan(double Number)
```

Source Code :

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415
int main(){
double Number, r;
Number = 1;
r = atan(Number);
    printf("Arc tangent of %.2lf = %.2lf in radians\n", Number, r);
r = atan(Number)*180/PI;   //radians to degree
    printf("Arc tangent of %.2lf = %.2lf in degree\n", Number, r);
return 0;
}
```

Output :

```
Arc tangent of 1.00 = 0.79 in radians
Arc tangent of 1.00 = 45.00 in degree
```

atanh() : Math Function

Arc hyperbolic tangent radians में return करता है |
Arc hyperbolic tangent में जो Number दिया जाता है वो -1 to +1 के बीच वाला होना चाहिए | पर -1 और +1 इन दो numbers को भी नहीं दिया जाता |

Syntax

```
double atanh(double Number)
```

Source Code :

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415
int main(){
double Number, r;
Number = 0.99;
r = atanh(Number);
    printf("Arc tangent of %.2lf = %.2lf in radians\n", Number, r);
r = atanh(Number)*180/PI;   //radians to degree
    printf("Arc tangent of %.2lf = %.2lf in degree\n", Number, r);
return 0;
}
```

Output :

```
Arc hyperbolic tangent of 0.99 = 2.65 in radians
Arc hyperbolic tangent of 0.99 = 151.65 in degree
```

atan2() : Math Function

Arc tangent2 radians में return करता है |

atan2() के लिए दो parameters होते है |
x-coordinatey
-coordinate

Syntax

```
double atan2(double y, double x)
```

Source Code :

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415
int main(){
double x, y, r;
x = -5;
y = 5;
r = atan2(y, x)*180/PI;   //radians to degree
   printf("Arc tangent2 of x = %.2lf, y = %.2lf is %.2lf degrees\n", x, y, r);
return 0;
}
```

Output :

```
Arc tangent2 of x = -5.00, y = 5.00 is 135.00 degrees
```

cbrt() : Math Function

Cube root के value को return करता है |

Syntax

```
double cbrt(double Number);
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number = 27;
   printf("Cube root of %.2lf is %.2lf", Number, cbrt(Number));
return 0;
}
```

Output :

```
Cube root of 27.00 is 3.00
```

ceil() : Math Function

ceil() में दिया हुआ Number अपने पासवाले या उससे बड़े या उसके बराबर के integer Number को return करता है |

Syntax

```
double ceil(double Number);
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number1 = 5.7;
double Number2 = 5.1;
double Number3 = 6;
    printf("ceil of %.2lf is %.2lf\n", Number1, ceil(Number1));
    printf("ceil of %.2lf is %.2lf\n", Number2, ceil(Number2));
    printf("ceil of %.2lf is %.2lf\n", Number3, ceil(Number3));
return 0;
}
```

Output :

```
ceil of 5.70 is 6.00
ceil of 5.10 is 6.00
ceil of 6.00 is 6.00
```

cos() : Math Function

cos() function में cosine का angle radians में return करता है |

Syntax

```
double cos(double Number);
```

Source Code :

```
#include <stdio.h>
#include <math.h>
#define PI 3.145
int main(){
double Number, r;
Number = 60;
r = cos(Number);
    printf("cosine of %.2lf = %.2lf in radians\n", Number, r);
r = cos(Number*PI/180);
    printf("cosine of %.2lf = %.2lf in degree\n", Number, r);
return 0;
}
```

Output :

```
cosine of 60.00 = -0.95 in radians
cosine of 60.00 = 0.50 in degree
```

cosh() : Math Function

Hyperbolic cosine का angle radians में return करता है |

Syntax

```
double cosh(double Number);
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double r;
r = cosh(0);
   printf("Hyperbolic cosine of 0 is %.2lf\n", r);
r = cosh(0.5);
   printf("Hyperbolic cosine of 0.5 is %.2lf\n", r);
r = cosh(1);
   printf("Hyperbolic cosine of 1 is %.2lf\n", r);
return 0;
}
```

Output :

```
Hyperbolic cosine of 0 is 1.00
Hyperbolic cosine of 0.5 is 1.13
Hyperbolic cosine of 1 is 1.54
```

exp() : Math Function

e को घातांक दिया जाता है | ये double data type की value return करता है |

Syntax

```
double exp(double Number);
```

जो number यहाँ पर दिया जाता है, वो e का घातांक होता है |

Note : e की value '2.71828183' होती है |

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double r;
r = exp(10);
   printf("e^10 = %lf", r);
return 0;
}
```

Output :

```
e^10 = 22026.465795
```

fabs() : Math Function

fabs() ये function absolute value को return करता है |

Syntax

```
double fabs(double Number);
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double r;
r = fabs(-4.5);
   printf("Abosolute value of -4.5 is %.2lf\n", r);
r = fabs(4.5);
   printf("Abosolute value of 4.5 is  %.2lf\n", r);
return 0;
}
```

Output :

```
Abosolute value of -4.5 is 4.50
Abosolute value of 4.5 is  4.50
```

floor() : Math Function

floor() ये function अपने पासवाले या दिए हुए number से छोटे या इससे बराबर के integer number को return करता है |

Syntax

```
double floor(double Number);
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double r;
r = floor(-4.5);
   printf("floor of -4.5 is %.2lf\n", r);
r = floor(-4.1);
   printf("floor of -4.1 is %.2lf\n", r);
r = floor(4.1);
   printf("floor of 4.1 is %.2lf\n", r);
r = floor(4.6);
   printf("floor of 4.6 is %.2lf\n", r);
r = floor(4.5);
   printf("floor of 4.5 is %.2lf\n", r);
return 0;
}
```

Output :

```
floor of -4.5 is -5.00
floor of -4.1 is -5.00
floor of 4.1 is 4.00
floor of 4.6 is 4.00
floor of 4.5 is 4.00
```

hypot() : Math Function

hypot() ये function 'Pythagoras Theorem' है |

Equation of Pythagoras Theorem

$c^2 = a^2 + b^2$

Syntax

```
double hypot(double a, double b);
```

$c^2 = 3^2 + 4^2$

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double a = 3.00, b = 4.00, r;
r = hypot(a, b);
    printf("Hypot a:%.2lf and b:%.2lf is %.2lf", a, b, r);
return 0;
}
```

Output :

```
Hypot a:3.00 and b:4.00 is 5.00
```

log() : Math Function

log() ये function Number का natural logarithm return करता है |

Syntax

```
double log(double Number);
```

$\log_e(1.5) = 0.41$

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number, r;
Number = 1.5;
r = log(Number);
   printf("Logarithm of %.2lf is %.2lf", Number, r);
return 0;
}
```

Output :

```
Logarithm of 1.50 is 0.41
```

log10() : Math Function

log10() ये function Number का base-10 logarithm return करता है |

Syntax

```
double log10(double Number);
```

$\log_{10}(1.5) = 0.18$

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number, r;
Number = 1.5;
r = log10(Number);
   printf("Base-10 Logarithm of %.2lf is %.2lf", Number, r);
return 0;
}
```

Output :

```
Logarithm of 1.50 is 0.41
```

pow() : Math Function

pow() ये function 'a' ये value होती है और 'b' ये उस value का घातांक होता है |

Syntax

```
double pow(double a, double b);
```

$a^b + 3^2 = 9$

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double a, b, r;
a = 3.00;
b = 2.00;
r = pow(a, b);
   printf("pow(%.2lf, %.2lf) = %.2lf", a, b, r);
return 0;
}
```

Output :

```
pow(3.00, 2.00) = 9.00
```

round() : Math Function

round() ये function Number की round value का integer return करता है |

Syntax

```
double round(double Number);
```

अगर 3.5 value हो तो round value 4.0 होता है | अगर value 3.1 और 3.4 के बीच में हो तो 3.0 round होता है |
अगर value 3.5 और 3.9 के बीच में हो तो 4.0 round होता है |

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double a, b, c, d, e, r;
a = 3.5;
b = 3.6;
c = 3.4;
d = 3.9;
e = 3.1;
r = round(a);
   printf("round of %.2lf is %.2lf\n", a, r);
r = round(b);
   printf("round of %.2lf is %.2lf\n", b, r);
r = round(c);
   printf("round of %.2lf is %.2lf\n", c, r);
r = round(d);
   printf("round of %.2lf is %.2lf\n", d, r);
r = round(e);
   printf("round of %.2lf is %.2lf\n", e, r);
return 0;
}
```

Output :

```
round of 3.50 is 4.00
round of 3.60 is 4.00
round of 3.40 is 3.00
round of 3.90 is 4.00
round of 3.10 is 3.00
```

sin() : Math Function

sin() ये function sine का angle radians में return करता है |

Syntax

```
double sin(double Number);
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number, r;
Number = 90.0;
r = sin(Number);
   printf("sine of %.2lf is %.2lf\n", Number, r);
r = sin(Number*PI/180);
   printf("sine of %.2lf is %.2lf in degrees\n", Number, r);
return 0;
}
```

Output :

```
sine of 90.00 is 0.89
sine of 90.00 is 1.00 in degrees
```

sqrt() : Math Function

sqrt() ये function Number का square root return करता है |

Syntax

```
double sqrt(double Number)
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number, r;
Number = 16;
r = sqrt(Number);
    printf("Square root of %.2lf is %.2lf \n", Number, r);
return 0;
}
```

Output :

```
Square root of 16.00 is 4.00
```

tan() : Math Function

tan() ये function Number का tangent का angle radians में return करता है |

Syntax

```
double tan(double Number)
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number, r;
Number = 6.5;
r = tan(Number);
    printf("tangent of %.2lf is %.2lf\n", Number, r);
Number = -6.5;
r = tan(Number);
    printf("tangent of %.2lf is %.2lf\n", Number, r);
return 0;
}
```

Output :

```
tangent of 6.50 is 0.22
tangent of -6.50 is -0.22
```

tanh() : Math Function

tanh() ये function Number का hyperboic tanhgent का angle radians में return करता है |

Syntax

```
double tanh(double Number)
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number, r;
Number = 0.6;
r = tanh(Number);
   printf("Hyperbolic tangent of %.2lf is %.2lf\n", Number, r);
Number = -0.6;
r = tanh(Number);
   printf("Hyperbolic tangent of %.2lf is %.2lf\n", Number, r);
return 0;
}
```

Output :

```
Hyperbolic tangent of 0.60 is 0.54
Hyperbolic tangent of -0.60 is -0.54
```

trunc() : Math Function

trunc() ये function Number को truncated integer Number में convert करता है |

Syntax

```
double trunc(double Number)
```

Source Code :

```
#include <stdio.h>
#include <math.h>
int main(){
double Number, r;
Number = 15.55;
r = trunc(Number);
   printf("Truncated of %.2lf is %.2lf\n", Number, r);
Number = -15.55;
r = trunc(Number);
   printf("Truncated of %.2lf is %.2lf\n", Number, r);
return 0;
}
```

Output :

```
Truncated of 15.55 is 15.00
Truncated of -15.55 is -15.00
```

ctype.h

Library Functions	Description
isalnum()	ये character alphanumeric है या नहीं ये check किया जाता है ।
isalpha()	ये character alphabetic है या नहीं ये check किया जाता है ।
iscntrl()	ये character control character है या नहीं ये check किया जाता है ।
isdigit()	ये character digit है या नहीं ये check किया जाता है ।
isgraph()	ये character graphical character है या नहीं ये check किया जाता है ।
islower()	ये character lowercase है या नहीं ये check किया जाता है ।
ispunct()	ये character puctuationn है या नहीं ये check किया जाता है ।
isspace()	ये character space है या नहीं ये check किया जाता है ।
isupper()	ये character uppercase है या नहीं ये check किया जाता है ।
isxdigit()	ये character hexadecimal है या नहीं ये check किया जाता है ।
tolower()	ये character uppercase से lowercase में convert किया जाता है ।
toupper()	ये character lowercase से uppercase में convert किया जाता है ।

isalnum : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(isalnum(i)){
    printf("%c is alphanumeric.\n", i);
}else{
    printf("%c is not alphanumeric.\n", i);
}
i = 'H';
if(isalnum(i)){
    printf("%c is alphanumeric.\n", i);
}else{
    printf("%c is not alphanumeric.\n", i);
}
i = '$';
if(isalnum(i)){
    printf("%c is alphanumeric.\n", i);
}else{
    printf("%c is not alphanumeric.\n", i);
}
return 0;
}
```

Output :

```
1 is alphanumeric.
H is alphanumeric.
$ is not alphanumeric.
```

isalpha : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(isalpha(i)){
    printf("%c is alphabetic.\n", i);
}else{
    printf("%c is not alphabetic.\n", i);
}
i = 'H';
if(isalpha(i)){
    printf("%c is alphabetic.\n", i);
}else{
    printf("%c is not alphabetic.\n", i);
}
i = '$';
if(isalpha(i)){
    printf("%c is alphabetic.\n", i);
}else{
    printf("%c is not alphabetic.\n", i);
}
return 0;
}
```

Output :

```
1 is not alphabetic.
H is alphabetic.
$ is not alphabetic.
```

iscntrl : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(iscntrl(i)){
    printf("%c is control character.\n", i);
}else{
    printf("%c is not control character.\n", i);
}
i = 'H';
if(iscntrl(i)){
    printf("%c is control character.\n", i);
}else{
    printf("%c is not control character.\n", i);
}
i = '$';
if(iscntrl(i)){
    printf("%c is control character.\n", i);
}else{
    printf("%c is not control character.\n", i);
}
i = '\t';
if(iscntrl(i)){
    printf("%c is control character.\n", i);
}else{
    printf("%c is not control character.\n", i);
}
return 0;
}
```

Output :

```
1 is not control character.
H is not control character.
$ is not control character.
        is control character.
```

isdigit : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(isdigit(i)){
    printf("%c is digit.\n", i);
}else{
    printf("%c is not digit.\n", i);
}
i = 'H';
if(isdigit(i)){
    printf("%c is digit.\n", i);
}else{
    printf("%c is not digit.\n", i);
}
i = '$';
if(isdigit(i)){
    printf("%c is digit.\n", i);
}else{
    printf("%c is not digit.\n", i);
}
return 0;
}
```

Output :

```
1 is digit.
H is not digit.
$ is not digit.
```

isgraph : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(isgraph(i)){
    printf("%c is graphical character.\n", i);
}else{
    printf("%c is not graphical character.\n", i);
}
i = 'H';
if(isgraph(i)){
    printf("%c is graphical character.\n", i);
}else{
    printf("%c is not graphical character.\n", i);
}
i = '$';
if(isgraph(i)){
    printf("%c is graphical character.\n", i);
}else{
    printf("%c is not graphical character.\n", i);
}
i = '\t';
if(isgraph(i)){
    printf("%c is graphical character.\n", i);
}else{
    printf("%c is not graphical character.\n", i);
}
return 0;
}
```

Output :

```
1 is graphical character.
H is graphical character.
$ is graphical character.
        is not graphical character.
```

islower : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(islower(i)){
    printf("%c is lowercase character.\n", i);
}else{
    printf("%c is not lowercase character.\n", i);
}
i = 'H';
if(islower(i)){
    printf("%c is lowercase character.\n", i);
}else{
    printf("%c is not lowercase character.\n", i);
}
i = 'h';
if(islower(i)){
    printf("%c is lowercase character.\n", i);
}else{
    printf("%c is not lowercase character.\n", i);
}
i = '$';
if(islower(i)){
    printf("%c is lowercase character.\n", i);
}else{
    printf("%c is not lowercase character.\n", i);
}
return 0;
}
```

Output :

```
1 is not lowercase character.
H is not lowercase character.
h is lowercase character.
$ is not lowercase character.
```

ispunct : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(ispunct(i)){
    printf("%c is punctuation.\n", i);
}else{
    printf("%c is not punctuation.\n", i);
}
i = 'H';
if(ispunct(i)){
    printf("%c is punctuation.\n", i);
}else{
    printf("%c is not punctuation.\n", i);
}
i = '"';
if(ispunct(i)){
    printf("%c is punctuation.\n", i);
}else{
    printf("%c is not punctuation.\n", i);
}
i = '$';
if(ispunct(i)){
    printf("%c is punctuation.\n", i);
}else{
    printf("%c is not punctuation.\n", i);
}
return 0;
}
```

Output :

```
1 is not punctuation.
H is not punctuation.
" is punctuation.
$ is punctuation.
```

isspace : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(isspace(i)){
    printf("%c is space.\n", i);
}else{
    printf("%c is not space.\n", i);
}
i = 'H';
if(isspace(i)){
    printf("%c is space.\n", i);
}else{
    printf("%c is not space.\n", i);
}
i = ' ';
if(isspace(i)){
    printf("%c is space.\n", i);
}else{
    printf("%c is not space.\n", i);
}
i = '$';
if(isspace(i)){
    printf("%c is space.\n", i);
}else{
    printf("%c is not space.\n", i);
}
return 0;
}
```

Output :

```
1 is not space.
H is not space.
  is space.
$ is not space.
```

isupper : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(isupper(i)){
    printf("%c is uppercase character.\n", i);
}else{
    printf("%c is not uppercase character.\n", i);
}
i = 'H';
if(isupper(i)){
    printf("%c is uppercase character.\n", i);
}else{
    printf("%c is not uppercase character.\n", i);
}
i = 'h';
if(isupper(i)){
    printf("%c is uppercase character.\n", i);
}else{
    printf("%c is not uppercase character.\n", i);
}
return 0;
}
```

Output :

```
1 is not uppercase character.
H is uppercase character.
h is not uppercase character.
```

isxdigit : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i;
i = '1';
if(isxdigit(i)){
   printf("%c is hexadecimal.\n", i);
}else{
   printf("%c is not hexadecimal.\n", i);
}
i = '9f';
if(isxdigit(i)){
   printf("%c is hexadecimal.\n", i);
}else{
   printf("%c is not hexadecimal.\n", i);
}
i = 'F';
if(isxdigit(i)){
   printf("%c is hexadecimal.\n", i);
}else{
   printf("%c is not hexadecimal.\n", i);
}
i = 'h';
if(isxdigit(i)){
   printf("%c is hexadecimal.\n", i);
}else{
   printf("%c is not hexadecimal.\n", i);
}
return 0;
}
```

Output :

```
1 is hexadecimal.
f is hexadecimal.
F is hexadecimal.
h is not hexadecimal.
```

tolower : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i, lower;
i = '1';
lower = tolower(i);
   printf("Converted in Lowercase : %c\n", lower);
i = 'H';
lower = tolower(i);
   printf("Converted in Lowercase : %c\n", lower);
i = 'h';
lower = tolower(i);
   printf("Converted in Lowercase : %c", lower);
return 0;
}
```

Output :

```
Converted in Lowercase : 1
Converted in Lowercase : h
Converted in Lowercase : h
```

toupper : Alphanumeric

Source Code :

```
#include <stdio.h>
#include <ctype.h>
int main(){
char i, upper;
i = '1';
upper = toupper(i);
   printf("Converted in uppercase : %c\n", upper);
i = 'H';
upper = toupper(i);
   printf("Converted in uppercase : %c\n", upper);
i = 'h';
upper = toupper(i);
   printf("Converted in uppercase : %c", upper);
return 0;
}
```

Output :

```
Converted in uppercase : 1
Converted in uppercase : H
Converted in uppercase : H
```

time.h

Library Functions	Description
asctime()	asctime() इस function और time के और कई function के लिए 'tm' नाम के structure को बनाया गया है ।
clock()	clock() ये system के time को return करता है ।
ctime()	ctime() ये funtion date और time को string के रूप से return करता है ।
difftime()	difftime() ये funtion दिए हुए दो समय का difference seconds में return करता है ।
gmtime()	gmtime() ये funtion से pointer से 'tm' structure को return किया जाता है ।
localtime()	localtime() ये funtion से pointer से 'tm' structure को return किया जाता है ।
mktime()	mktime() ये funtion date और time को broken-down time('tm' structure)में convert कर देता है ।

asctime() - Time Function

asctime() ये Calendar time का एक format है |

Syntax

```
char *asctime(const struct tm *time_ptr);
```

asctime() इस function और time के और कई function के लिए 'tm' नाम के structure को बनाया गया है | जो pointer string data type को return करता है |

'tm' structure in time.h(Codeblock)

```
struct tm
{
int tm_sec; /* Seconds: 0-59 (K&R says 0-61?) */
int tm_min; /* Minutes: 0-59 */
int tm_hour; /* Hours since midnight: 0-23 */
int tm_mday; /* Day of the month: 1-31 */
int tm_mon; /* Months *since* january: 0-11 */
int tm_year; /* Years since 1900 */
int tm_wday; /* Days since Sunday (0-6) */
int tm_yday; /* Days since Jan. 1: 0-365 */
int tm_isdst; /* +1 Daylight Savings Time, 0 No DST,
* -1 don't know */
};
```

tm नाम के structure में second, minute, hour और कई Members या elements बनाये गए है | जिससे time को अच्छी तरीके से पढा जा सकता है |

User को सिर्फ 'tm' structure के variable को declare करना पड़ता है |

Format of Calendar Time
Www Mmm dd hh:mm:ss yyyy
Sat Dec 31 23:59:59 2017\n\0

आखिर में asctime() Newline(\n) और NULL Character(\0) return करता है |

Source Code :

```
#include <stdio.h>
#include <time.h>
int main(){
struct tm t;
t.tm_sec  = 59;   // seconds
t.tm_min  = 59;   // minutes
t.tm_hour = 23;   // hours
t.tm_mday = 31;   // day of month
t.tm_mon  = 11;   // Months
t.tm_year = 116; // Years  (1900+116)
t.tm_wday = 6;    // Numbers of Weeks
   printf(asctime(&t));
return 0;
}
```

Output :

```
Sat Dec 31 23:59:59 2016
```

clock() - Time Function

clock() ये system के time को return करता है |

Syntax

```
clock_t clock();
```

clock() function clock ticks को return करता करता है | अगर उसे second में convert करना हो तो CLOCKS_PER_SEC से भाग देना पडता है |
clock() function Program Execution के लिए जो समय बिता है, उसे बताता है |
function अगर कोई कार्य नहीं करता तो '-1' return करता है |

Source Code :

```
#include <stdio.h>
#include <time.h>
int main(){
int i;
clock_t time1, time2, time3;
time1 = clock();
printf("start ticks = %ld\n", time1);
for(i=0; i< 1000; i++){
    printf("\n");
}
time2 = clock();
    printf("end ticks = %ld\n", time2);
time3 =(time2 - time1) / CLOCKS_PER_SEC;
    printf("Total Time in seconds: %ld", time3);
return 0;
}
```

Output :

```
start ticks = 0




end ticks = 3087
Total Time in seconds: 3
```

ctime() - Time Function

ctime() ये funtion date और time को string के रूप से return करता है |

Syntax

```
char *ctime(const time_t * time_ptr);
```

निचे दिए हुए format से string return होता है |
Www Mmm dd hh:mm:ss yyyy
ये function '\n'(Newline) और '\0'(NULL Character) के साथ string return होता है |

Source Code :

```
#include <stdio.h>
#include <time.h>
int main ()
{
time_t t;
time(&t);
    printf("Current time is %s", ctime(&t));
return 0;
}
```

Output :

```
Current time is Sat Dec 31 18:28:00 2016
```

difftime() - Time Function

difftime() ये funtion दिए हुए दो समय का difference seconds में return करता है |

Syntax

```
double difftime(time_t time2, time_t time1);
```

Source Code :

```
#include <stdio.h>
#include <time.h>
int main (){
time_t time1,time2;
double diff_t;
time(&time1);
    printf("Press any key\n");
getche();
    printf("\n");
time(&time2);
diff_t = difftime (time2,time1);
    printf ("You got this value in %ld seconds.", diff_t );
return 0;
}
```

Output :

```
Current time is Sat Dec 31 18:28:00 2016
```

difftime() - Time Function

gmtime() ये funtion से pointer से 'tm' structure को return किया जाता है |

Syntax

```
struct tm *gmtime(const time_t *timeptr);
```

Structure में Greenwich Mean Time(GMT) की value होती है जो timeptr इस pointer से point की जाती है |

Source Code :

```
#include <stdio.h>
#include <time.h>
#define UTC (0)
#define CET (+1)
int main (){
time_t now;
struct tm *mptr;
time (&now);
mptr = gmtime (&now);
  printf ("Iceland : %2d:%02d\n", (mptr->tm_hour+UTC)%24, mptr->tm_min);
  printf ("Germany : %2d:%02d\n", (mptr->tm_hour+CET)%24, mptr->tm_min);
return 0;
}
```

Output :

```
Iceland : 17:19
Germany : 18:19
```

localtime() - Time Function

localtime() ये funtion से pointer से 'tm' structure को return किया जाता है |

Syntax

```
struct tm *localtime(const time_t *timeptr);
```

tm नाम के structure में date और time के members होते है |

Source Code :

```
#include <stdio.h>
#include <time.h>
int main(){
time_t now;
struct tm *mptr;
time(&now);
mptr = localtime(&now);
    printf ("Current time and date : %s", asctime(mptr));
return 0;
}
```

Output :

```
Current time and date : Sat Dec 31 23:05:43 2016
```

localtime() - Time Function

mktime() ये funtion date और time को broken-down time('tm' structure)में convert कर देता है और Pointer से Calendar time को return करता है |
अगर function नाकाम हो जाए तो '-1' return होता है |

Syntax

```
time_t mktime(struct tm *timeptr);
```

Source Code :

```
#include <stdio.h>
#include <time.h>
int main(){
time_t tformat;
struct tm t;
t.tm_year = 2017-1900;
t.tm_mon = 0;
t.tm_mday = 1;
t.tm_wday = 0;
t.tm_hour = 0;
t.tm_min = 0;
t.tm_sec = 0;
t.tm_isdst = 0;
tformat = mktime(&t);
    printf(ctime(&tformat));
return 0;
}
```

Output :

```
Sun Jan 01 00:00:00 2017
```

string.h

Library Functions	Description
strcat()	एक String से दूसरे String को जोड़ा जाता है ।
strchr()	दिए हुए string से एक character का पहला occurrence के आगे का string pointer को return करता है ।
strcmp()	दो String को Compare किया जाता है । ये case-sensetive है ।
strcmpi()	दो String को Compare किया जाता है । ये case-sensetive नहीं है ।
strcpy()	एक string को दूसरे string में copy करता है ।
strdup()	String का duplicate बनाता है ।
strlen()	String की Length निकाली जाती है ।
strlwr()	Uppercase के Characters को Lowercase में convert किया जाता है ।
strncat()	दिए हुए number के जितने character है उनसे String को जोड़ा जाता है ।
strncpy()	दिए हुए number के जितने character एक string से दूसरे string में copy किया जाता है ।
strnset()	दिए हुए number और दिए हुए character के हिसाब से string को replace करता है ।
strrchr()	दिए हुए string से एक character का आखिरी occurrence के आगे का string pointer को return करता है ।
strrev()	String को उलटी दिशा से print करता है ।
strrstr()	दिए हुए String का आखिरी string occurrence के आगे का string pointer को return करता है ।
strset()	दिए हुए character से पूरे string को replace करता है ।
strstr()	दिए हुए String का पहला string occurrence के आगे का string pointer को return करता है ।
strupr()	Lowercase के Characters को Uppercase में convert किया जाता है ।

strcat() - String Function

strcat ये String का एक Function है | एक String से दूसरे String को जोड़ा जाता है |

Syntax

```
strcat(destination_string, source_string);
```

Destination_string - ये वो parameter है जिसके साथ source का string जोड़ा जाता है | String के आखिर में null character (\0) होता है | Source string destination string के साथ जुड़ते समय उसको remove कर देता है |
Source_string - ये वो parameter है जिसका string destination string के साथ आखिर में जोड़ा जाता है |

किसी integer value को एक variable को दूसरे variable से जोड़ना हो तो arithmetic operator (+) का use नहीं कर सकते |

for e.g.
int num1 = 5 ;
int num2 = 5 ;
int num3 = num1 + num2 ;

इनका output 55 मिलना चाहिए लेकिन इनका output 10 मिलेगा |किसी char को भी एक दूसरे से arithmetic operators के साथ नहीं जोड़ा जा सकता |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main (){
char str1[20] = "Welcome";
char str2[20] = " Friend";
strcat(str1, str2);
    printf("Concatenation String : %s", str1);
return 0;
}
```

Output :

```
Concatenation String : Welcome Friend
```

strcat() - String Function

दिए हुए string से एक character का पहला occurrence के आगे का string pointer को return करता है |

Syntax

```
strchr(string, int character);
```

string - ये एक normal string है |
i**nt character** - ये दिए हुए string में से दिए हुए character का पहला occurrence के आगे का string pointer को return करता है |

अगर दिया हुआ character string को नहीं मिलता तो वो NULL character return करता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main (){
char str[] = "Hello Friends";
char *ptr;
ptr = strchr(str, 'F');
    printf("%s", ptr);
return(0);
}
```

Output :

```
Friends
```

strcmp() - String Function

दो String को Compare किया जाता है | ये case-sensetive है |

Syntax

```
strcmp(string1, string2);
```

string1 -ये वो String है जिसके साथ String2 को Compare किया जाता है |
string2 - ये वो String है जिसके साथ String1 को Compare किया जाता है |

अगर दोनों string एक जैसे होते है तो ये '0' return करता है |अगर दोनों string अलग-अलग होते है तो '1' या '-1' return करता है |

Note : ये strcmp() function case-sensitive है | इसमें 'h' और 'H' ये दोनों अलग-अलग है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[] = "Hello" ;
char str2[] = "World" ;
int a = strcmp(str1, str2) ;
   printf ("%d\n", a) ;
int b = strcmp(str1, "Hello") ;
   printf ("%d\n", b) ;
int c = strcmp(str1, "hello") ; // strcmp is case-sensitive
   printf ("%d\n", c) ;

return 0;
}
```

Output :

```
-1
0
-1
```

strcmpi() - String Function

दो String को Compare किया जाता है | ये case-sensetive नहीं है |

Syntax

```
strcmpi(string1, string2);
```

string1 - ये वो String है जिसके साथ String2 को Compare किया जाता है |
string2 - ये वो String है जिसके साथ String1 को Compare किया जाता है |

अगर दोनों string एक जैसे होते है तो ये '0' return करता है |अगर दोनों string अलग-अलग होते है तो '1' या '-1' return करता है |

Note : ये strcmpi() function case-sensitive है | इसमें 'h' और 'H' ये एक ही अलग-अलग है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[] = "Hello" ;
char str2[] = "World" ;
int a = strcmpi(str1, str2) ;
   printf ("%d\n", a) ;
int b = strcmpi(str1, "Hello") ;
   printf ("%d\n", b) ;
int c = strcmpi(str1, "hello") ; // strcmpi is not case-sensitive
   printf ("%d\n", c) ;

return 0;
}
```

Output :

```
-1
0
0
```

strcpy() - String Function

दो String को Compare किया जाता है | ये case-sensetive नहीं है |

Syntax

```
strcpy(destination_string, source_string);
```

destination_string - ये वो parameter है जिसपर source के string की value copy की जाती है | अगर destination string पर कोई value भी हो तो वो overwrite हो जाती है |
source_string - ये वो parameter है जिसकी value destination पर copy की जाती है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[20] = "Welcome";
char str2[20] = "Friend";
   printf("Value of str2 = %s\n", str2 );
strcpy(str2, str1);
   printf("Copy str1 to str2 = %s", str2 );
return 0;
}
```

Output :

```
Value of str2 = Friend
Copy str1 to str2 = Welcome
```

strdup() - String Function

String का duplicate बनाता है |

Syntax

```
strdup(string);
```

string - ये String है जिसका duplicate बनाया जाता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[] = "Hello World";
char *ptr;
ptr = strdup(str);
    printf("Duplicate string : %s", ptr);
return 0;
}
```

Output :

```
Duplicate string : Hello World
```

strlen() - String Function

String की Length निकाली जाती है |

Syntax

```
strlen(string);
```

string - ये एक normal string है, जिसकी length निकली जाती है |

strlen से निकला हुआ output integer value ही होती है ,ये किसी दूसरे integer variable में भी store करके रख सकते है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[30] = "Hello World";
int length ;
   printf("String : %s\n", str);
length = strlen(str);
   printf("Length of str is %d", length);
return 0;
}
```

Output :

```
String : Hello World
Length of str is 11
```

strlwr() - String Function

Uppercase के Characters को Lowercase में convert किया जाता है |

Syntax

```
strlwr(string);
```

string - ये वो string है जिसको lowercase में convert किया जाता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[50] = "HELLO WORLD";
int lwr ;
    printf("String :  %s\n", str );
    printf("Lowercase of str = %s", strlwr(str));
return 0;
}
```

Output :

```
String :  HELLO WORLD
Lowercase of str = hello world
```

strncat() - String Function

दिए हुए number के जितने character है उनसे String को जोड़ा जाता है |

Syntax

```
strncat(destination_string, source_string, size_t num);
```

destination_string - ये वो string जिसके साथ source string को जोड़ा जाता है |
source_string - ये वो string जिसके साथ destination string को बाद में जोड़ा जाता है |
size_t num - यहाँ पर जो integer value दी जाती है उतने character वो source string से लेता है |

Program में strncat(str1, str2, 2); ऐसा लिखा है , इसका मतलब ये है कि, ये 2 characters str2 से लेगा |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[50] = "HELLO WORLD";
int lwr ;
    printf("String :  %s\n", str );
    printf("Lowercase of str = %s", strlwr(str));
return 0;
}
```

Output :

```
String :  HELLO WORLD
Lowercase of str = hello world
```

strncpy() - String Function

दिए हुए number के जितने character एक string से दूसरे string में copy किया जाता है |

Syntax

```
strncpy(destination_string, source_string, size_t num);
```

destination_string - ये वो parameter है जिसपर source के string की value copy की जाती है | अगर destination string पर कोई value भी हो तो वो overwrite हो जाती है |
source_string - ये वो parameter है जिसकी value destination पर copy की जाती है |
size_t num - यहाँ पर जो integer value दी जाती है उतने character वो destination string से लेकर source string पर copy कर देता है |

Program में strncpy(str1, str2, 2); ऐसा लिखा है , इसका मतलब ये है कि, ये 2 characters str2 से लेगा |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[20] = "Welcome";
char str2[20] = "Friend";
   printf("Value of str2 = %s\n", str2 );
strncpy(str2, str1, 2);
   printf("Copy str1 to str2 = %s", str2 );
return 0;
}
```

Output :

```
Value of str2 = Friend
Copy str1 to str2 = Weiend
```

strnset() - String Function

दिए हुए number और दिए हुए character के हिसाब से string को replace करता है |

Syntax

```
strnset(string, char ch, int c);
```

destination_string - ये एक normal string है |
char ch - ये वो character है जिससे string के हर character को replace किया जाता है |
i**nt c** - यहाँ पर जितना number है उतने character string से replace किया जाते है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[20] = "Hello World";
   printf("Value of str = %s\n", str);
strnset(str, '.', 2);
   printf("%s", str);
return 0;
}
```

Output :

```
Value of str = Hello World
..llo World
```

strrchr() - String Function

दिए हुए string से एक character का आखिरी occurrence के आगे का string pointer को return करता है |

Syntax

```
strrchr(string, int character);
```

string - ये एक normal string है |
int character - ये वो character है जिसका आखिरी occurrence के आगे का string pointer को return किया जाता है |

अगर strrchr() function को कोई character नहीं मिलता तो वो NULL character return करता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[20] = "Hello World";
char *ptr;
ptr = strrchr(str, 'o');
    printf("%s", ptr);
return 0;
}
```

Output :

```
orld
```

strrev() - String Function

String को उलटी दिशा से print करता है |

Syntax

```
strrev(string);
```

string - ये एक normal string है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[20] = "Hello World";
strrev(str);
    printf("%s", str);
return 0;
}
```

Output :

```
dlroW olleH
```

strrstr() - String Function

दिए हुए String का आखिरी string occurrence के आगे का string pointer को return करता है |

Syntax

```
strrstr(string1, string2);
```

string1 - ये एक normal string है |
string2 - string1 में से ये string find करके उसका आखिरी occurrence pointer को return करता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[100] = "Hi How are you? Hi I am Fine";
char *ptr;
ptr = strrstr(str, "Hi");
    printf("%s", ptr);
return 0;
}
```

Output :

```
Hi I am Fine
```

strset() - String Function

दिए हुए String का आखिरी string occurrence के आगे का string pointer को return करता है |

Syntax

```
strset(string, int character);
```

string - ये एक normal string है |
int character - ये एक-एक character करके सभी characters को replace कर देता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[20] = "Hi How are you? Hi I am Fine";
strset(str, '$');
    printf("%s", str);
return 0;
}
```

Output :

```
$$$$$$$$$$$$$$$$$$$$
```

strstr() - String Function

दिए हुए String का आखिरी string occurrence के आगे का string pointer को return करता है |

Syntax

```
strstr(string1, string2);
```

string - ये एक normal string है |
int character - string1 में से ये string find करके उसका पहला occurrence pointer को return करता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[100] = "Hi How are you? Hi I am Fine";
char *ptr;
ptr = strstr(str, "Hi");
   printf("%s", ptr);
return 0;
}
```

Output :

```
Hi How are you? Hi I am Fine
```

strupr() - String Function

Lowercase के Characters को Uppercase में convert किया जाता है |

Syntax

```
strupr(string);
```

string - ये वो string है जिसको Uppercase में convert किया जाता है |

Source Code :

```
#include <stdio.h>
#include <string.h>
int main(){
char str[50] = "hello world";
int lwr ;
    printf("String :  %s\n", str );
    printf("Uppercase of str = %s", strupr(str));
return 0;
}
```

Output :

```
String :  hello world
Uppercase of str = HELLO WORLD
```