AMBITION GURU

Programming Techniques Using C

I semester, BCA

Ambition Guru College
2026

Syllabus

AMBITION GURU

Unit	Contents	Hours	Remarks
1.	Introduction of Programming Concept	4	
2.	**Introduction to C**	**6**	
3.	Control Structure	7	
4.	Function	6	
5.	Array, Pointer and String	7	
6.	Structure and Union	7	
7.	Input Output and File handling	5	
8.	Introduction to Graphics	3	

Practical Works   Credit hours : 3

AMBITION GURU

Unit II

(6 hrs.)

UNIT II: Introduction to C

- History of C Program, Basic Structure of C Program, Character set, Token, Keywords and Comments, Variables and Constants, Rules for naming and declaring data variables,

- Data Types, Type Conversion, Input and Output,

- Operators: Arithmetic, Relational, Logical, Increment/Decrement, Assignment, Bitwise, Ternary/ conditional(:?), Comma

AMBITION GURU

Unit II

(6 hrs.)

Section 1: History & Foundation of C Programming

Section 2: Tokens, Keywords, Comments, Variables & Constants

Section 3: Data Types, Type Conversion, Input and Output

Section 4: C Operators Explained

AMBITION GURU

Section 1: History & Foundation of C Programming
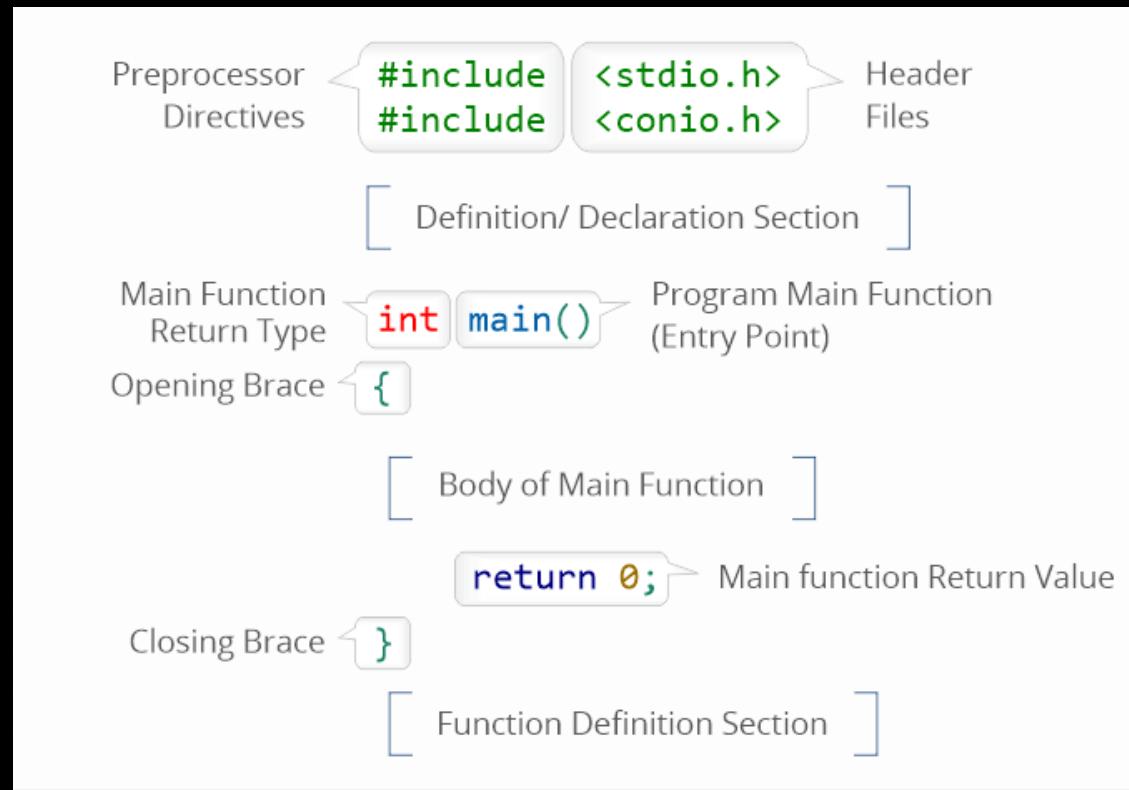
- Brief History of C

- Basic Structure of a C Program

- Character Set in C

History of C

- **Origins and Precursors (1960s-1970s):** C evolved from BCPL (Basic Combined Programming Language) and B, developed by Martin Richards and Ken Thompson, respectively, as simplified languages for system programming.

- **Creation of C (1972):** Dennis Ritchie at Bell Labs developed C to rewrite the Unix operating system, combining high-level functionality with low-level control, making it portable and efficient.

- **Unix and Adoption (1973-1980):** The Unix OS was rewritten in C, showcasing its portability and performance, leading to widespread adoption in academia, research, and industry.

- **Standardization (1989):** ANSI standardized C as "ANSI C" (C89), followed by international recognition (ISO C). Updates like C99, C11, and C17 introduced modern features, enhancing functionality and safety.

- **Legacy and Impact**: C became the foundation for languages like C++, Java, and Python, and remains pivotal in operating systems, embedded systems, and performance-critical applications.

Basic Structure of C program

Basic Structure of C program: Key Components

Key Components of structure of C program:

- **Preprocessor Directives**

- **Main function**

- **Body of main function**

- **Opening and closing braces**

Basic Structure of C program : Key Components

AMBITION GURU

Preprocessor Directives:

- These lines start with #include, which is a preprocessor directive.

- They include header files like <stdio.h> (standard input/output functions) and <conio.h> (console input/output, mainly used in Turbo C/C++).

- This section is also called the Definition/Declaration Section.

Main function:

- This is the entry point of every C program.

- int is the return type, meaning the function returns an integer value to the operating system.

- main() is where program execution begins.

Basic Structure of C program : Key Components

Body of Main Function:

- The logic of the program goes here.

- return 0; signals that the program ended successfully

Opening and closing of braces:

- Marks the **beginning of the main function body**.

- Marks the end of the main function.

- Encloses the body of the function, completing the Function Definition Section

Character Set in C

- Character set is a package of valid characters recognized by compiler/interpreter.

- C uses character as building blocks to form basic program elements.(E.g Constants, variables, expressions etc.

- The characters available in C are categorized into following types:

1. **Letters/Alphabets :** Uppercase(A-Z), Lowercase(a-z)

2. **Digits** (0-9)

3. **Special symbols** (+,-,*, ! ,%,_,<,>, :)

4. **Whitespace characters (** Blank Space, Newline, Horizontal tab, Vertical tab)

12

AMBITION GURU
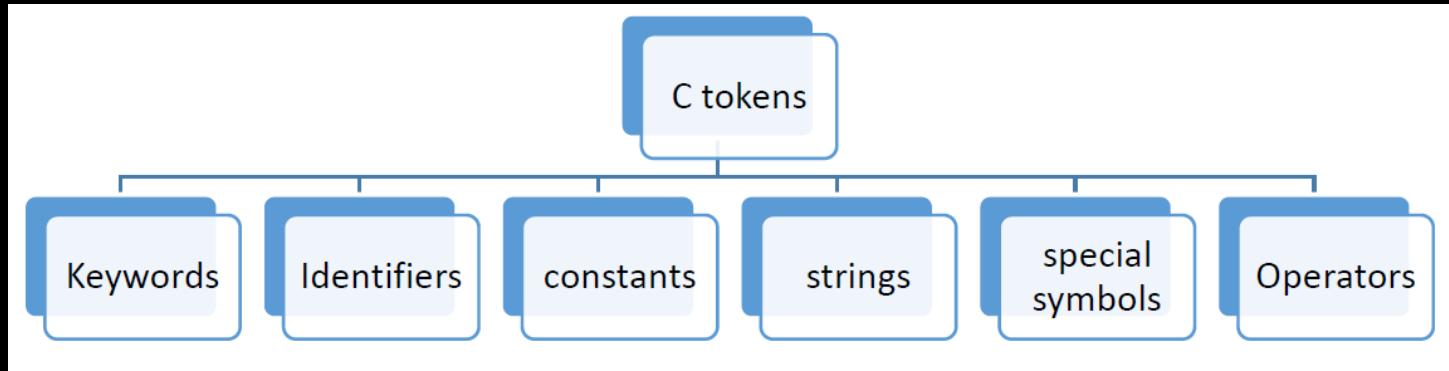
Section 2: Tokens, Keywords, Comments, Variables & Constants

- What are Tokens?: Keywords, Identifiers, Constants, Strings, Operators

- Keywords in C

- Comments

- Variables and Constants

- Rules for Naming Variables

AMBITION GURU

Tokens

Tokens

- Tokens are the smallest individual units of program.

- C has six types of tokens . They are as follows:

C tokens

Keywords | Identifiers | constants | strings | special symbols | Operators

Keywords

Keywords

- Keywords are predefined words whose meaning has already defined to c compilers. These keywords are used for pre-defined purposes and cannot be used as identifier.

- The standard keywords are:

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

AMBITION GURU

Identifiers

Identifiers

- Identifiers are the names given by user to various program elements such as variables, function and arrays.

  <u>Rules of naming Identifiers</u>

  - First character must be alphabet (or underscore).

  - Must consists of only letters, digits or underscore

  - Keyword cannot be used.

  - The length of identifiers should not be greater than 31 characters.

  - Must not contain whitespace.

  - Identifiers are case-sensitive

Identifiers

- Identifiers are the names given by user to various program elements such as variables, function and arrays.

Rules of naming Identifiers

- First character must be alphabet (or underscore).

- Must consists of only letters, digits or underscore

- Keyword cannot be used.

- The length of identifiers should not be greater than 31 characters.

- Must not contain whitespace.

- Identifiers are case-sensitive

AMBITION GURU

Variables

Variables

- Variable is defined as a meaningful name given to the data storage location in computer memory.

- It is a medium to store and manipulate data. The value of variable can change during execution of program.

Variables

AMBITION GURU

Variable declaration

Naming a variable along with its data type for programming is known as variable declaration.

The declaration deals with two things:

- It tells the compiler what the variable name is
- It specifies what type of data the variable can hold

Syntax:

```
data_type variable_name ;
```

eg. `int roll;`

roll is a variable of type integer. roll can only store integer variable only.

Variables

AMBITION GURU

Rules for declaring variables

- Variable name must begin with a letter, some system permit underscore as first character.

- ANSI standard recognizes a length of 31 characters.

- It should not be keyword.

- Whitespace is not allowed.

- Variable name are case sensitive .ie. uppercase and lowercase are different. Thus variable temp is not same as TEMP.

- No two variables of the same name are allowed to be declared in the same scope.
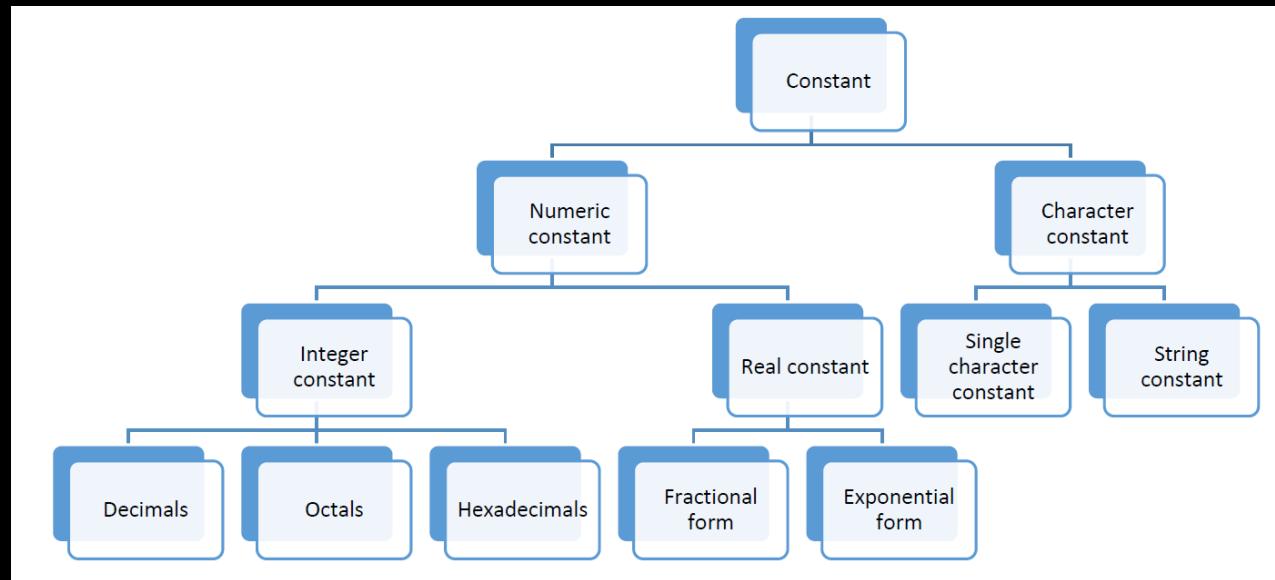
AMBITION GURU

Constants

Constants

- Constant refers to fixed values that do not change during the execution of program. These fixed values are also called literals. The C constants are as shown in figure.
- Number associated constant are numeric constant and character associated constant are character constants.

Constants

1) Numeric Constant
It consist of:

1.1 Integer constant
It refers to sequence of digits.

There are three types of integer constants.

- **Decimals:** Decimal Integer consist of set of digits, 0 through 9, preceded by an optional +ve or – ve sign.eg.124,-321,678, 654343 etc.

- **Octal:** An octal Integer constant consist of any combination of digits from the set 0 to 7, with a leading 0. eg 037, 075, 0664, 0551

- **Hexadecimal:** Hexadecimal are sequence of digits 0-9 and alphabet A-F with preceding 0x or 0X. eg. 0x5567, 0X53A, 0xFF etc.

Constants

1.2. Real or floating point constant

They are numeric constant with decimal points. The real constant are further categorized as

- **Fractional real constant:** They are the set of digits from 0 to 9 with decimal points.eg.394.7867,-0.78676

- **Exponential real constant:** In the exponential form the constants are represented in following form: *mantissa e exponent*
    where, the mantissa is either a real number expressed in decimal notation or an integer but exponent is always in integer form.

eg.
$21565.32=2.156532E4$, Where $E4=10^4$
Similarly, -0.000000368 is equivalent to -3.68E-7

Constants

2) Character Constant

Single Character Constant

A single character constant (or simply character constant) contains a single character enclosed

within a pair of single quote marks. Eg.

'5' 'X' 'A' '4' etc.

String constant

A string constant is a sequence of characters enclosed in double quotes. The characters may be

numbers, special characters and blank space.eg. "Hello" "green" "305" etc.

29

AMBITION GURU

Comments

Comments

AMBITION GURU

Comments are non-executable lines in a C program used to **explain the code**, make it more **readable**, and help **debug or maintain** it later. The compiler **ignores** comments when the program runs.

1. **Single-line Comment**
Starts with //

Used for short notes or explanations

Code:

```
// This is a single-line comment
int x = 10; // Initialize variable x
```

2. Multi-line Comment
Starts with /* and ends with */

Used for longer explanations or to comment out blocks of code.

Code:

```
/* This is a multi-line comment
   It can span several lines */
int y = 20;
```

31

AMBITION GURU

Section 3: Data Types, Type Conversion, Input and Output

- Data Types in C

- Type Conversion

- Input & Output in C

AMBITION GURU

Data types

Data Types

Data type is a classification of type of data that a variable can hold in programming. The data type specifies the range of values that can be used and size of memory needed for that value.

Generally data type can be categorized as:

1. Primary

2. Derived

3. User defined

Data Types

AMBITION GURU

1. <u>Primary data type</u>: These are the basic data types provided by C.

Data type	Description	Required Memory	Range	Format specifier
char	Used to store character value Eg. 'a','c', 'x'	1 byte	-128 to 127	%c
int	Used to store whole numbers/non fractional numbers. Eg.101, 205, -908	2 bytes	-32768 to +32767	%d
float	Used to store fractional number and has precision of 6 digits. Eg. 5.674, 304.67,67.8903	4 bytes	$-3.4*10^{38}$ to $3.4*10^{38}$	%f
double	Used when precision of float is insufficient and it has precision of 14 digits.	8 bytes	$-1.7*10^{308}$ to $1.7* 10^{308}$	%lf
void	Has no values and used as return type for function that do not return a value. Eg. void main()			

Data Types

1. <u>Primary data type</u>: These are the basic data types provided by C.

Data Type	Description	Example
`int`	Integer values (whole numbers)	`int age = 25;`
`float`	Floating-point numbers (single precision)	`float weight = 70.5;`
`double`	Double-precision floating-point numbers	`double pi = 3.14159;`
`char`	Single character	`char grade = 'A';`
`void`	No value (used for functions that return nothing)	`void display();`

Data Types

2. Derived data type: They are derived from existing primary data types. E.g.

Array, Union, Structure

3. User defined data type
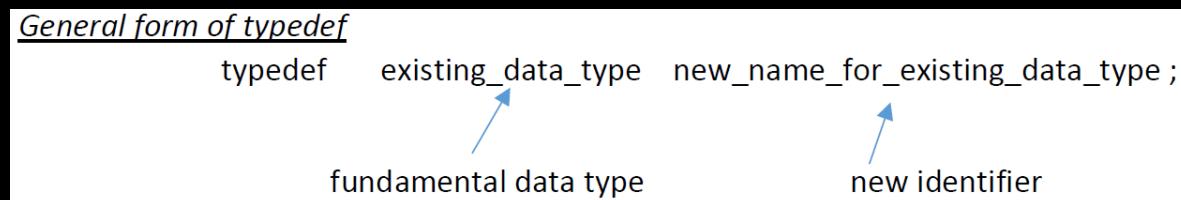
The data type that are defined by user is known as user defined data types.

Examples: **enum** (Enumerated data type) and **typedef** (type definition datatype)

General form of typedef

      typedef   existing_data_type   new_name_for_existing_data_type ;

                          fundamental data type   new identifier

---

Example: typedef int integer ;
integer symbolizes int data type Now, we can declare int variable "a" as **integer a** rather than **int a**

AMBITION GURU

Type Casting

Type Casting

- Converting an expression of a given type into another type is known as type casting.

- Here, it is best practice to convert lower data type to higher data type to avoid data loss.

1. Implicit Conversion

- Implicit conversions do not require any operator for conversion. They are automatically performed when a value is copied to a compatible data type in the program.

- Here, the value of `i` has been promoted from **int** to **float** and we have not had to specify any type-casting operator.

Type Casting

Program

```c
#include<stdio.h>
int main()
{
int i=20;
float result;
result=i; // implicit conversion
printf("value=%f", result);
return 0;
}
```

Output
Result=20.000000

Type Casting

AMBITION GURU

2) Explicit conversion

In this type of conversion one data type is converted forcefully to another data type by the user.

The general rule for cast is

```
(type-name) expression
```

Where type-name is one of the standard c data types. The expression may be constant , variable or an expression.

Type Casting

2) Explicit conversion

Example	Action
x=(int)7.5	7.5 is converted to integer by truncation
a=(int)21.3/(int)4.5	Evaluated as 21/4 and result would be 5
b=(double)sum/n	Division is done in floating point mode
z=(int)a+b	a is converted into integer and then added to b

Type Casting

AMBITION GURU

2) Explicit conversion

Program
```c
#include<stdio.h>
#include<conio.h>
int main()
{
int a=5,b=2;
float result;
result=(float)a/b; // Explict conversion
printf("value=%f",result);
getch();
return 0;
}
```

Output
Result=2.50000

Input/ Output functions

Input/ Output functions

C programming language provides many built-in functions to read any given input and to display data on screen when there is need to output the result. The header file for input/output functions is `<stdio.h>`

The input/output function is classified into two types.
1) Formatted I/O functions
2) Unformatted I/O functions

AMBITION GURU

Compiled by ab

AMBITION GURU

Formatted I/O functions

Input/ Output functions

1) Formatted I/O functions

Formatted Input
- Formatted Input refers to an input data that can be arranged in a particular format according to user requirements. The built-in function scanf() can be used to input data into the computer from standard Input device.

The general form of scanf is
```
scanf("control string",arg1,arg2 ….argn);
```

- The control string refers to field in which data is to be entered.
- The arguments arg1,arg2 ,…argn specify the address of locations where data is stored.

eg. `scanf("%d%d",&num1,&num2);`

Input/ Output functions

Formatted output

* Formatted output functions are used to display data in a particular specified format. The printf() is an example of formatted output function.

* The general form of printf() statement is

    `printf("control string",arg1,arg2 …… argn);`

The control string may consist of the following data items.
1. Characters that will be printed on the screen as they appear.
2. Format specifications that define the output format for display of each item.
3. Escape sequence characters such as \n,\t and \b

The arguments arg1,arg2,…..argn are the variables whose value are formatted and printed according to specifications of the control string.

Eg. `printf("First number=%d\tSecond number=%d",num1,num2);`

48

Input/ Output functions

Program to read multiple values of different data types using single scanf() function

```c
#include<stdio.h>
#include<conio.h>
int main()
{
    char name[20];
    int age;
    char gender;
    float weight;
    printf("Enter your Name,Age,Gender and Weight\n");
    scanf("%s %d %c %f",name,&age,&gender,&weight);
    printf("Your Name=%s\n",name);
    printf("Your Age=%d\n",age);
    printf("Your Gender=%c\n",gender);
    printf("Your Weight=%0.2f",weight);
    getch();
    return 0;
}
```

AMBITION GURU

Unformatted I/O functions

Input/ Output functions

Unformatted I/O functions

- Unformatted I/O function do not allow the user to read or display data in a desired format.

- These type of library functions basically deal with a single character or string of characters.

- The functions getchar(), putchar(),gets(),puts() are considered as unformatted functions.

51

Input/ Output functions

Unformatted I/O functions

i) getchar() and putchar()
The getchar() reads a character from a standard input device. It buffers the input until the 'ENTER' key is pressed and then assigns this character to character variable.

   Syntax is : **`character_variable=getchar();`**
where character_variable refers to some previously declared character variable.
Eg.
char c;
c=getchar();

putchar() function displays a character to standard output device.

   Syntax is: `putchar(character_variable);`

Input/ Output functions

Unformatted I/O functions

Program

```
#include<stdio.h>

#include<conio.h>

int main()

{

char ch;

printf("Enter the character");

ch=getchar();

printf("Entered characer is :");

putchar(ch);

getch();

return 0;

}
```

Input/ Output functions

Unformatted I/O functions

gets() and puts()

- The gets() function is used to read a string of text containing whitespaces, until newline character is encountered. It can be used as an alternative function for reading strings.

- Unlike scanf() functions, it does not skip whitespaces(ie.It can be used to read multiwords string)

Syntax: `gets(string_variable);`

The puts() function is used to display the string on the screen.

Syntax: `puts(string_variable);`

Input/ Output functions

Unformatted I/O functions

Program

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

AMBITION GURU

Section 4: C Operators Explained

Operators: Arithmetic, Relational,

Logical, Increment/Decrement, Assignment,

Bitwise, Ternary/ conditional(:?), Comma

Operators

- Operator is a symbol that tells the computer to perform certain mathematical or logical manipulations.

- Operators are used in program to manipulate data and variables.

Operand

The data items on which operators are act upon are called operands.

e.g. a + b

- Here, symbol + is known as operator

- a and b are operands

Operators

AMBITION GURU

Expression

The combination of variables, constants and operators written according to syntax of programming language is known as Expression.

e.g. a + b * c - 5

Types of operators

Types of Operators are classified as:

1. On the basis of no of operands

2. On the basis of function/utility

On the basis of no of operands

Operators

AMBITION GURU

1. On the basis of number of operands

i) Unary operator
The operator which operates on single operand known as unary operator.
(++) increment Operator
(--) Decrement operator
+ Unary plus
- Unary minus

eg . ++x, -5, +8 , y- - etc

Operators

1. On the basis of number of operands

ii) Binary operator

The operator which operates on two operands are known as binary operator.

e.g + (addition), − (subtraction) , / (division) , * (multiplication) , <(less than)

>(greater than)

etc.

eg 3+3, 5>2 , 6*7 , 15-4

Operators

AMBITION GURU

1. On the basis of number of operands

iii) Ternary operators

The Operators that operates on three operands are known as ternary operator. Ternary operator pair "?:" is available in c .

Syntax:

```
expression1? expression2:expression3
```

The expression1 is evaluated first,

* if it is true then expression2 is evaluated and its value becomes value of the expression
* If it is false expression3 is evaluated and its value becomes value of the expression.

E.g .Let us consider the program statement

```
int a=15,b=10,max;
max = (a>b)?a: b;
```

Here, The value of a will assigned to max.

On the basis of no of utility/functions

Operators

On the basics of utility /functions

1) Arithmetic Operators

They are used to perform Arithmetic/Mathematical operations on operands.

Operator	Meaning	Example If a=20 and b=10
+	Addition	a+b=30
-	Subtraction	a-b=10
*	Multiplication	a*b=200
/	Division	a/b=2
%	Modulo Division	a%b=0

Operators: Arithmetic Operators

WAP to illustrate Arithmetic Operators in C

AMBITION GURU

```c
#include <stdio.h>

int main() {
    int num1, num2;
    int sum, difference, product, remainder;
    float quotient;

    // Input two numbers
    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    // Arithmetic operations
    sum = num1 + num2;
    difference = num1 - num2;
    product = num1 * num2;
    quotient = (float)num1 / num2;
    remainder = num1 % num2;

    // Display the results
    printf("\nResults of Arithmetic Operations:\n");
    printf("Addition: %d + %d = %d\n", num1, num2, sum);
    printf("Subtraction: %d - %d = %d\n", num1, num2, difference);
    printf("Multiplication: %d * %d = %d\n", num1, num2, product);
    printf("Division: %d / %d = %.2f\n", num1, num2, quotient);
    printf("Modulus: %d %% %d = %d\n", num1, num2, remainder);

    return 0;
}
```

Operators

2) Relational Operators

These are used to compare two quantities and depending on their relation take certain decision. The value of relational operator is either 1(if the condition is true) and 0 (if the condition is false).

Operator	Meaning	Example If a=20 and b=10
==	Equal to	(a==b) is not True
!=	Not Equal to	(a!=b) is True
>	Greater than	(a>b) is True
<	Less than	(a<b) is not True
>=	Greater than or Equal to	(a>=b) is True
<=	Less than or Equal to	(a<=b) is not True

Operators

AMBITION GURU

3) Logical Operators

Logical Operators are used to compare or evaluate logical and relational expressions and returns either 0 or 1 depending upon whether expressions results true or false.

Operator	Meaning	Example If a=20 and b=10
&&	Logical AND	Expression ((a==b)&&(a>15)) equals to 0
||	Logical OR	Expression ((a==b)||(a>15)) equals to 1
!	Logical NOT	Expression !(a==b) equals to 1

Operators

AMBITION GURU

4) Assignment Operators

Assignment Operators are used to assign the result of an expression to a variable. The mostly used assignment operator is "=".

Operator	Name	Example
=	Assignment	c=a+b will assign value of a+b into c
+=	Add AND assignment	c+=a is equivalent to c=c+a
-=	Subtract AND assignment	c-=a is equivalent to c=c-a
*=	Multiply AND assignment	c*=a is equivalent to c=c*a
/=	Divide AND assignment	c/=a is equivalent to c=c/a
%=	Modulus AND assignment	c%=a is equivalent to c=c%a

Operators

AMBITION GURU

5) Increment and Decrement Operator

The increment operators (++) causes its operand to be increased by 1 whereas decrement operator (--) causes its operand to be decreased by 1.

They are unary operators (ie. They operate on single operand).It can be written as

- x++ or ++x (post and pre increment)
- x-- or - - x (post and pre decrement)

Note: The increment and decrement operators can be used as postfix and prefix notations.

Prefix notation: The variable is incremented (or decremented) first and then expression is evaluated using the new value of the variable.

Operators

5) Increment and Decrement Operator

Eg. int m=5;

y=++m;

In this case the value of y and m would be 6.

Postfix notation: The expression is evaluated first using the original value of the variable and then variable is incremented (or decremented by one).

E.g. int m=5;

y=m++;

The value of y would be 5 and m would be 6

Operators

6) Conditional Operator

The ternary Operator pair "?:" is available in c to construct conditional expressions of the form.

`expression1? expression2: expression3`

The expression1 is evaluated first,

- if it is true then the expression 2 is evaluated and its value becomes the value of the expression.

- If expression1 is false expression3 is evaluated and its value becomes the value of the expression.

Operators

6) Conditional Operator

Program:

```c
#include <stdio.h>

int main() {
    int a, b, max;
    a = 10;
    b = 15;
    max = (a > b) ? a : b;

    printf("Value of a: %d\n", a);
    printf("Value of b: %d\n", b);
    printf("The maximum value is: %d\n", max);
    return 0;
}
```

Operators:

7) Bitwise operators

The operators which are used for the manipulation of data at bit level. These operators are used for testing the bits, shifting them right or left.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right

Operators: Bitwise Operators

The truth tables for &, |, and ^ is as follows

P	Q	p & q	p	q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Assume a = 60 and b = 13 in binary format, they will be as follows:

a = 0011 1100

b = 0000 1101

----

a&b   =   0000 1100 =>12

a|b   =   0011 1101 =>61

a^b   =   0011 0001 => 49

Operators: Bitwise Operators

AMBITION GURU

Bitwise shift operators
i. Left shift <<
This causes the operand to be shifted left by some bit positions.
General form:
Operand<<n
Eg.
n=60, n1=n<<3

                 N 0000 0000 0011 1100
First shift: 0000 0000 0111 1000
Second shift: 0000 0000 1111 0000
Third shift: 0000 0001 1110 0000

After left shifting value of n1 becomes 480

Operators: Bitwise Operators

Bitwise shift operators

ii. Right shift >>

This causes operand to be shifted to the right by some bit positions.

Operand >>n

Eg. n=60, n2=n>>3

N   0000 0000 0011 1100

First shift   0000 0000 0001 1110

Second shift   0000 0000 0000 1111

Third shift   0000 0000 0000 0111

After Right shifting value of n2 becomes 7

Operators:

8) Special operators

C supports some special operators. Some of them are

i. Sizeof operator

The sizeof is a compile time operator and when used with an operand, it returns the number of bytes the operand occupies. The operand may be variable, constant or data type qualifier.

Examples:

    m=sizeof(sum);

    n=sizeof(long int);

Operators:

Compiled by ab

Program

```c
#include<stdio.h>

#include<conio.h>


int main()

{

float num;

printf("Number of bytes allocated =%d",sizeof(num));

getch();

return 0;

}
```

Operators:

ii. Comma operator

The comma operator can be used to link related expressions together .A comma-linked list are evaluated left to right and the value of right most expression is the value of combined expression.

For example the statement

    value = (x=10, y=5, x+y);

First assigns the value 10 to x then assigns 5 to y and finally assigns 15 (i.e. 10 + 5) to value.

Operator precedence and associativity

- Operator precedence is a predefined rule of priority of operators. If more than one operators are involved in an expression the operator of higher precedence is evaluated first.

- Associativity indicates the order in which multiple operators of same precedence executes.

Precedence	Operator	Associativity
1	(), {}	Left to right
2	++, --, !	Right to left
3	*, /, %	Left to right
4	+, -	Left to right
5	<, <=, >, >=	Left to right
6	== ,!=	Left to right
7	&&	Left to right
8	||	Left to right
9	?:	Right to left
10	=, *=, /=, %=, -=	Right to left

Some Questions to Work On:

Q.1. Debug the following program:

```c
#include <stdio.h>
int main() {
    int a, b;
    printf("Enter two numbers: ")
    scanf("%d %d", a, b);
    result = a + b;
    printf("The sum of %d and %d is %d\n", a, b result);
    return 0
}
```

Some Questions to Work On:

Q.1. Debug the following program. (Solution)

- Missing semicolon after the printf statement.

- scanf is missing the address-of operator (&).

- The variable result is used but not declared.

- Missing comma between format specifiers in printf.

- Missing semicolon at the end of return statement.

Some Questions to Work On:

AMBITION GURU

1. WAP to calculate area and circumference of circle having the radius r should be taken from user.

2. WAP to swap two numbers.

3. WAP to find the reverse of a given (3 digit) number.

AMBITION GURU

1. WAP to calculate area and circumference of circle having the radius r should be taken from user.

```c
#include<stdio.h>
#include<conio.h>
int main()
{
float r,area,circum;
printf("Enter the radious of circle");
scanf("%f",&r);
area=3.1416*r*r;
circum=2*3.1416*r;
printf("\nArea of circle=%f",area);
printf("\nCircumference of circle is %f",circum);
getch();
return 0;
}
```

AMBITION GURU

2. WAP to swap two numbers.

```c
#include<stdio.h>
#include<conio.h>
int main()
{
int a,b,temp;
a=5;
b=10;
printf("Number before swapping a=%d and b=%d",a,b);
temp=a;
a=b;
b=temp;
printf("\nNumber after swapping a=%d and b=%d",a,b);
getch();
return 0;
}
```

AMBITION GURU

3. WAP to find the reverse of a
given (3 digit) number

```c
#include<stdio.h>
#include<conio.h>
int main()
{
int num,a,b,c,rev;
printf("Enter the number");
scanf("%d",&num);
c=num%10;
b=(num/10)%10;
a=num/100;
rev=c*100+b*10+a;
printf("Reverse of given number is %d",rev);
getch();
return 0;
}
```

AMBITION GURU

Questions to practice

Questions

1. Write in brief about history in C.
2. List different types of operators and explain any four of them.
3. Explain the basic structure of C Programming.
4. Describe different formatted input and output functions. Why do we use them?
5. Define data types. Explain different types of data types in brief.
6. What is variable? How is it different from constant? How do you write comments in c?
7. Define keyword. Write down rules for defining variables.
8. Explain unformatted I/O functions in detail.
9. Define explicit conversion. Write down a code to show explicit conversion.
10. WAP to swap two numbers.
11. WAP to calculate area and circumference of circle having the radius r should be taken from user.
12. WAP to find the reverse of a given (3 digit) number.
13. Write short notes:
    a. Conditional Operator
    b. Bitwise Operator
    c. Implicit Conversion

AMBITION GURU

AMBITION GURU

THANK YOU
Any Queries ?