**HTML**

HTML stands for Hyper Text Markup Language, which is the most widely used language on Web to develop web pages. HTML was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

**What is HTML?**

HTML is the standard markup language for creating Web pages.

* HTML stands for **H**yper **T**ext **M**arkup **L**anguage
* HTML describes the structure of Web pages using markup
* HTML elements are the building blocks of HTML pages
* HTML elements are represented by tags
* HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
* Browsers do not display the HTML tags, but use them to render the content of the page

## HTML Tags

HTML tags are element names surrounded by angle brackets:

**<tagname>content here...</tagname>**

* HTML tags normally come in pairs like <p> and </p>
* The first tag in a pair is the start tag, the second tag is the end tag
* The end tag is written like the start tag, but with a forward slash inserted before the tag name

## The <!DOCTYPE> Declaration

The <!DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.

It must only appear once, at the top of the page (before any HTML tags).

The <!DOCTYPE> declaration is not case sensitive.

The <!DOCTYPE> declaration for HTML5 is: **<!DOCTYPE html>**

**HTML Tags Chart**

To use any of the following HTML tags, simply select the HTML code you'd like and copy and paste it into your web page.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Name** | **Code Example** | **Browser View** |
| <!-- | **comment** | **<!--**This can be viewed in the **HTML** part of a document**-->** | Nothing will show |
| <a - | **anchor** | **<a href=**"http://www.google.co"> Visit Our Site**</a>** | Visit google |
| <b> | **bold** | **<b>**Example**</b>** | **Example** |
| <big> | **big (text)** | **<big>**Example**</big>** | Example |
| <body> | **body of HTML**  **document** | **<body>**The content of your **HTML** page**</body>** | Contents of your web page |
| <br> | **line break** | Keonics**<br>**Hubli | Keonics  Hubli |
| <center> | **center** | **<center>**This will center your contents**</center>** | This will center your contents |
| <dd> | **definition description** | <dl>  <dt>Definition Term</dt>  **<dd>Definition of the term</dd>**  <dt>Definition Term</dt>  **<dd>Definition of the term</dd>**  </dl> | Definition Term  **Definition of the term**  Definition Term  **Definition of the term** |
| <dl> | **definition list** |
| <dt> | **definition term** |
| <em> | **emphasis** | This is an **<em>**Example**</em>** of using the emphasis tag | This is an *Example* of using the emphasis tag |
| <font> | **font** | **<font** face="Times New Roman" size="3" color="#ff0000">Example**</font>** | Example |
| <h1>  <h2>  <h3>  <h4>  <h5>  <h6> | **heading 1**  **heading 2**  **heading 3**  **heading 4**  **heading 5**  **heading 6** | **<h1>**Heading 1 Example**</h1>**  **<h2>**Heading 2 Example**</h2>**  **<h3>**Heading 3 Example**</h3>**  **<h4>**Heading 4 Example**</h4>**  **<h5>**Heading 5 Example**</h5>**  **<h6>**Heading 6 Example**</h6>** |  |
| <hr> | **horizontal rule** | **<hr />** | Contents of your web page  Contents of your web page |
| <hr> | **horizontal rule** | **<hr** width="50%" size="3" /> | Contents of your web page  Contents of your web page |
| <hr> | **horizonta l rule** | **<hr** width="50%" size="3" noshade /> | Contents of your web page  Contents of your web page |
| <hr> (Internet Explorer) | **horizonta l rule** | **<hr** width="75%" color="#ff0000" size="4"  /> | Contents of your web page  Contents of your web page |
| <html> | **hypertext markup language** | **<html>**  <head>  <meta>  <title>Title of your web page</title>  </head>  <body>**HTML web page** contents  </body>  **</html>** | Contents of your web page |
| <i> | **italic** | **<i>**Example**</i>** | *Example* |
| <img> | **image** | **<img** src="Earth.gif" width="41" height="41" border="0" alt="text describing the image" /> |  |

**Html Programs**

**1)**

<html>

<head>

<title>

My First web page

</title>

</head>

<body>

Welcome to HTML!...

</body>

</html>

**2)**

<html>

<head>

<title>

back color

</title>

</head>

<body bgcolor="wheat">

Welcome to HTML!...

</body>

</html>

**3)**

<html>

<head>

<title>

text color

</title>

</head>

<body text="red">

Welcome to HTML!...

</body>

</html>

**4)**

<html>

<head>

<title>

heading tags

</title>

</head>

<body>

<h1>Heading 1</h1>

<h2>Heading 2</h2>

<h3>Heading 3</h3>

<h4>Heading 4</h4>

<h5>Heading 5</h5>

<h6>Heading 6</h6>

</body>

</html>

**5)**

<html>

<head>

<title>

font size tags

</title>

</head>

<body>

<font size="1">Font size 1</font><br>

<font size="2">Font size 2</font><br>

<font size="3">Font size 3</font><br>

<font size="4">Font size 4</font><br>

<font size="5">Font size 5</font><br>

<font size="6">Font size 6</font><br>

<font size="7">Font size 7</font>

</body>

</html>

**6)**

<html>

<head>

<title>

iframe

</title>

</head>

<body>

<h1>KEONICS</h1>

<iframe src="29.html" align="right" height="500" width="180"></iframe>

<iframe src="25.html" align="bottom" align="left" height="200" width="300">

</body>

</html>

**7)**

<html>

<head>

<title>

iframe

</title>

</head>

<body>

<h1>KEONICS</h1>

<iframe src="29.html" align="right" height="500" width="180"></iframe>

<iframe src="25.html" align="bottom" align="left" height="200" width="300">

</body>

</html>

**8)**

<html>

<head>

<title>

Image

</title>

</head>

<body>

<img src="5.jpg" height="300" width="400" border="4">

</body>

</html>

**9)**

<html>

<head>

<title>

iframe

</title>

</head>

<body>

<h1>KEONICS</h1>

<iframe src="29.html" align="right" height="500" width="180"></iframe>

<iframe src="25.html" align="bottom" align="left" height="200" width="300">

</body>

</html>

**10)**

<html>

<head>

<title>

use of horizontal line

</title>

</head>

<body>

<h1>This is heading 1</h1>

<p>This is some text.</p>

<hr>

<h2>This is heading 2</h2>

<p>This is some other text.</p>

<hr>

<h2>This is heading 3</h2>

<p>This is some other text.</p>

</body>

</html>

**11)**

<html>

<head>

<title>

Paragraph with alignment tags

</title>

</head>

<body>

<p align="right">Paragraph 1</p>

<p align="center">Paragraph 2</p>

<p align="left">Paragraph 3</p>

</body>

</html>

**12)**

<html>

<head>

<title>

order listing

</title>

</head>

<body>

<H2>ORDERED LIST</H2>

<ol>

<li>C</li>

<li>C++</li>

<li>Java</li>

</ol>

<ol type="I">

<li>C</li>

<li>C++</li>

<li>Java</li>

</ol>

<ol type="i">

<li>C</li>

<li>C++</li>

<li>Java</li>

</ol>

<ol type="A">

<li>C</li>

<li>C++</li>

<li>Java</li>

</ol>

<ol type="a">

<li>C</li>

<li>C++</li>

<li>Java</li>

</ol>

</body>

</html>

**13)**

<html>

<head>

<title>

Unorder listing

</title>

</head>

<body>

<H2>UNORDERED LIST</H2>

<ul>

<li>C</li>

<li>C++</li>

<li>Java</li>

</ul>

<ul type="square">

<li>C</li>

<li>C++</li>

<li>Java</li>

</ul>

<ul type="circle">

<li>C</li>

<li>C++</li>

<li>Java</li>

</ul>

<ul type="disc">

<li>C</li>

<li>C++</li>

<li>Java</li>

</ul>

</body>

</html>

**14)**

<html>

<head>

<title>definition list</title>

</head>

<body>

<h2>Definition List</h2>

<dl>

<dt>COMPUTER</dt>

<dd>Computer is an electronic device accepts data from

user and process the data ,gives you required result as output

</dd>

</dt>

</dl>

<dl>

<dt>What is HTML?</dt>

<dd>HTML is the standard markup language for creating Web pages.

</dd>

</dt>

</dl>

</body>

</html>

**15)**

<html>

<head>

<title>

MOVING text

</title>

</head>

<body>

<marquee direction="right" bgcolor="orange">

<h1>KEONICS COMPUTER CENTRE</h1>

</marquee>

</body>

</html>

**16)**

<html>

<head>

<title>

Login page

</title>

</head>

<body>

<table border="1" bgcolor="pink">

<tr>

<th colspan="2">Login Up </th>

</tr>

<tr>

<th>User Name</th>

<td><input type="text"></td>

</tr>

<tr>

<th>Password</th>

<td><input type="password" maxlength="8"></td>

</tr>

<tr>

<tr>

<td colspan="2" align="center">

<input type="Submit" value="Login">

<input type="Reset" value="Clear">

</td>

</tr>

</table>

</body>

</html>

**17)**

<html>

<head>

<title>

Sign up page

</title>

</head>

<body>

<table border="1" bgcolor="pink">

<tr>

<th colspan="2">Sign Up </th>

</tr>

<tr>

<th>First Name</th>

<td><input type="text"></td>

</tr>

<tr>

<th>Middle Name</th>

<td><input type="text"></td>

</tr>

<tr>

<th>Last Name</th>

<td><input type="text"></td>

</tr>

<tr>

<th>Address</th>

<td><TextArea>hubli</TextArea></td>

</tr>

<tr>

<th>Gender</th>

<td>

<input type="radio" name="gr1">Male

<input type="radio" name="gr1">Female

</td>

</tr>

<tr>

<th>Language Known</th>

<td>

<input type="Checkbox">Kannada <br>

<input type="Checkbox">Hindi<br>

<input type="Checkbox">English<br>

</td>

</tr>

<tr>

<th>Select Country</th>

<td>

<Select>

<option>India</option>

<option>Aus</option>

<option>England</option>

<option>USA</option>

</Select>

</td>

</tr>

<tr>

<td colspan="2" align="center">

<input type="Submit" value="Save">

<input type="Reset" value="Clear">

</td>

</tr>

</table>

</body>

</html>

**18)**

<frameset cols="20%,20%,30%,30%">

<frame src="29.html">

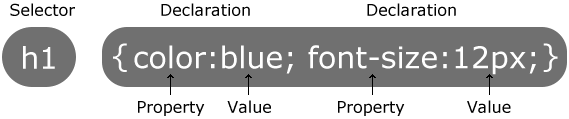
<frame src="36.html">

<frame src="25.html">

<frame src="26.html">

</frameset>

**What is CSS?**

* **CSS** stands for **C**ascading **S**tyle **S**heets
* CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
* CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
* External style sheets are stored in **CSS files**
* CSS Syntax
* A CSS rule-set consists of a selector and a declaration block:
* 
* The selector points to the HTML element you want to style.
* The declaration block contains one or more declarations separated by semicolons.
* Each declaration includes a CSS property name and a value, separated by a colon.
* A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.
* In the following example all <p> elements will be center-aligned, with a red text color:

**The element Selector**

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

Example

p {  
    text-align: center;  
    color: red;  
}

**The id Selector**

The id selector uses the id attribute of an HTML element to select a specific element. The id of an element should be unique within a page, so the id selector is used to select one unique element! To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The style rule below will be applied to the HTML element with id="para1":

Example

#para1 {  
    text-align: center;  
    color: red;  
}

**The class Selector**

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with class="center" will be red and center-aligned:

Example

.center {  
    text-align: center;  
    color: red;  
}

**Grouping Selectors**

If you have elements with the same style definitions, like this:

h1 {  
    text-align: center;  
    color: red;  
}  
  
h2 {  
    text-align: center;  
    color: red;  
}

p {

text-align: center;

    color: red;  
}

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

In the example below we have grouped the selectors from the code above:

Example

h1, h2, p {  
    text-align: center;  
    color: red;  
}

**CSS Comments**

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with /\* and ends with \*/. Comments can also span multiple lines:

Example

p {  
    color: red;  
    /\* This is a single-line comment \*/  
    text-align: center;  
}  
  
/\* This is  
a multi-line  
comment \*/

**Three Ways to Insert CSS**

There are three ways of inserting a style sheet:

* External style sheet
* Internal style sheet
* Inline style

**External Style Sheet**

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

Example

<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a **.css extension.**

Here is how the "**mystyle.css"** looks:

body {  
    background-color: lightblue;  
}  
h1 {  
    color: navy;  
    margin-left: 20px;  
}

**Internal Style Sheet**

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

Example

<head>  
<style>  
body {  
    background-color: linen;  
}  
  
h1 {  
    color: maroon;  
    margin-left: 40px;  
}   
</style>  
</head>

**Inline Styles**

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

Example

<h1 style="color:blue;margin-left:30px;">This is a heading</h1>

**CSS Comments**

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers.

A CSS comment starts with /\* and ends with \*/. Comments can also span multiple lines:

Example

p {  
    color: red;  
    /\* This is a single-line comment \*/  
    text-align: center;  
}  
  
/\* This is  
a multi-line  
comment \*/

**CSS Programs**

**1)**

<html>

<body>

<h2 style="background-color:red">

Red background-color

</h2>

<h2 style="background-color:green">

Green background-color

</h2>

<h2 style="background-color:blue;color:FFFFFF">

Blue background-color and white text color

</h2>

<h2 style="background-color:orange">

Orange background-color

</h2>

<h2 style="background-color:yellow">

Yellow background-color

</h2>

<h2 style="background-color:CYAN">

Cyan background-color

</h2>

<h2 style="background-color:black;color:white">

Black background-color and white text color

</h2>

</body>

</html>

**2)**

<html>

<head>

<style>

body {

background-image: url("1.jpg");

background-repeat:repeat;

background-position: left top;

background-attachment:fixed;

}

</style>

</head>

<body>

<h1>Hello World!</h1>

<p>The background-image is fixed. Try to scroll down the page.</p>

<p>copy paragraph 20 times for scrolling</p>

</body>

</html>

**3)**

<html>

<head>

<style>

p.dotted {border-style: dotted;}

p.dashed {border-style: dashed;}

p.solid {border-style: solid;}

p.double {border-style: double;}

p.groove {border-style: groove;}

p.ridge {border-style: ridge;}

p.inset {border-style: inset;}

p.outset {border-style: outset;}

p.none {border-style: none;}

p.hidden {border-style: hidden;}

p.mix {border-style: dotted dashed solid double;}

</style>

</head>

<body>

<h2>The border-style Property</h2>

<p>This property specifies what kind of border to display:</p>

<p class="dotted">A dotted border.</p>

<p class="dashed">A dashed border.</p>

<p class="solid">A solid border.</p>

<p class="double">A double border.</p>

<p class="groove">A groove border.</p>

<p class="ridge">A ridge border.</p>

<p class="inset">An inset border.</p>

<p class="outset">An outset border.</p>

<p class="none">No border.</p>

<p class="hidden">A hidden border.</p>

<p class="mix">A mixed border.</p>

</body>

</html>

**4)**

<html>

<head>

<style>

div {

background-color: lightgrey;

width: 300px;

border: 25px solid green;

padding: 25px;

margin: 50px;

}

</style>

</head>

<body>

<h2>Demonstrating the Box Model</h2>

<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>

<div>This text is the actual content of the box. We have added a 25px padding, 25px margin and a 25px green border. </div>

</body>

</html>

**5)**

<html>

<body>

<h3 style="color:Tomato;">Hello World</h3>

<p style="color:Blue;">You can set the color of text</p>

<p style="color:Green;">You can set the color of text</p>

</body>

</html>

**6)**

<html>

<head>

<style>

p.normal {

font-weight: normal;

font-family: IMPACT;

font-size: 250%;

font-style: italic;

}

p.light {

font-weight: lighter;

font-size: 300%;

font-style: oblique;

}

p.thick {

font-weight: bold;

font-size: 40px;

font-variant: small-caps;

}

p.thicker {

font-weight: 900;

}

</style>

</head>

<body>

<p class="normal">This is a paragraph. normal</p>

<p class="light">This is a paragraph. light</p>

<p class="thick">This is a paragraph . thick</p>

<p class="thicker">This is a paragraph. thicker</p>

</body>

</html>

**7)**

<html>

<head>

<style>

/\* unvisited link \*/

a:link {

color: red;

}

/\* visited link \*/

a:visited {

color: green;

}

/\* mouse over link \*/

a:hover {

color: hotpink;

}

/\* selected link \*/

a:active {

color: blue;

}

</style>

</head>

<body>

<p><b><a href="f1.html" >This is a link</a></b></p>

<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>

<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>

</body>

</html>

**8)**

<html>

<head>

<style>

h1

{

color:Crimson ; /\* any color value\*/

letter-spacing:3px; /\* any pixel value\*/

word-spacing:5px; /\* any pixel value \*/

text-align:center; /\* left center right justify \*/

text-decoration:underline; /\* overline line-through \*/

text-transform:capitalize; /\* uppercase lowercase \*/

text-shadow: 3px 3px Gold ;

/\* horizontal\_shadow vertical\_shadow color \*/

}

h2

{

text-indent: 25px;

direction:ltr; /\* rtl \*/

}

</style>

</head>

<body>

<h1>KEONICS COMPUTER TRAINING CENTRE</h1>

<h2> IT PARK HUBLI</h2>

</body>

</html>

**9)**

<html>

<body>

<h1 style="border: 2px solid Tomato;">Hello World</h1>

<h1 style="border: 2px solid DodgerBlue;">Hello World</h1>

<h1 style="border: 2px solid Violet;">Hello World</h1>

</body>

</html>

**10)**

<html>

<head>

<style>

p { display: block; }

h1 { display:inline; }

</style>

</head>

<body>

<p>Hello Javatpoint</p>

<p>Java </p>

<h1>SQL </h1>

<h1>HTML </h1>

<h1>CSS </h1>

</body>

</html>

**JavaScript - Overview**

**What is JavaScript ?**

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

**Client-Side JavaScript**

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

**Advantages of JavaScript**

The merits of using JavaScript are −

* **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
* **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

**JavaScript - Syntax**

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

<script ...>

JavaScript code

</script>

The script tag takes two important attributes −

* **Language** − This attribute specifies what scripting language you are using.
* **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to **"text/javascript".**

<script language = "javascript" type = "text/javascript">

JavaScript code

</script>

## First JavaScript Code

<html>

<body>

<script language = "javascript" type = "text/javascript">

document.write("Hello World!")

</script>

</body>

</html>

## Comments in JavaScript

JavaScript supports both C-style and C++-style comments, Thus −

* Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
* Any text between the characters /\* and \*/ is treated as a comment. This may span multiple lines.
* JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
* The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

## JavaScript Variables

* Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.
* Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

<script type = "text/javascript">

var name;

var address;

</script>

**Note** − Use the **var** keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

## JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

* **Global Variables** − A global variable has global scope which means it can be defined anywhere in your JavaScript code.
* **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

## JavaScript Variable Names

While naming your variables in JavaScript, keep the following rules in mind.

* You should not use any of the JavaScript reserved keywords as a variable name.
* JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, **123test** is an invalid variable name but **\_123test** is a valid one.
* JavaScript variable names are case-sensitive. For example, **Name** and **name** are two different variables.

# JavaScript - Operators

JavaScript supports the following types of operators.

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators

**Java script Programs**

1. <HTML>

<body>

<p> SUM OF TWO NUMBERS</P>

<script type="text/javascript">

var a,b;

a=10;

b=20;

var sum=a+b; document.write("a+b="+sum);

</script>

</body>

</HTML>

**2)**

<HTML>

<body>

<p> ARITHMATIC OPERATION</P>

<script type="text/javascript">

var a,b;

a=20;

b=3;

var sum=a+b;

var d=a-b;

var p=a\*b;

var q=a/b;

var r=a%b; document.write("</h3>a+b="+sum+"</br>");

document.write("a-b="+d+"</br>"); document.write("a\*b="+p+"</br>"); document.write("a/b="+q+"</br>"); document.write("a%b="+r+"</br></h3>");

</script>

</body>

</HTML>

**3)**

<!--program to illustrate do while -->

<html>

<body>

<script type = "text/javascript">

var count = 0;

document.write("Starting Loop" + "<br />");

do {

document.write("Current Count : " + count + "<br />");

count++;

}

while (count < 5);

document.write ("Loop stopped!");

</script>

<p>Set the variable to different value and then try...</p>

</body>

</html>

4)

<html>

<body>

<script type = "text/javascript">

var n=-5;

if(n>0)

{

document.write("<br> Given no "+n+" is +ve no")

}

if(n<0)

{

document.write("<br> Given no "+n+" is -ve no")

}

</script>

</body>

</html>

5)

<html>

<body>

<script type = "text/javascript">

var n1=3,n2=4;

if(n1>n2)

{

document.write("<br>"+n1+" is big")

}

else

{

document.write("<br>"+n2+" is big")

}

</script>

</body>

</html>

6)

<html>

<body>

<script type = "text/javascript">

var area,l=2,b=2,r=2,pi=3.142,h=2;

var ch=2;

document.write("<br>1-->area of a traingle");

document.write("<br>2-->area of a rectangle");

document.write("<br>3-->area of a circle") ;

document.write("<br>Your choice is 2:");

switch(ch)

{

case 1:

area=0.5\*b\*h;

document.write("<br>Area of a traingle="+area);

break;

case 2:

area=l\*b;

document.write("<br>Area of a rectangle="+area);

break;

case 3:

area=pi\*r\*r;

document.write("<br>Area of a circle="+area);

break;

default:

document.write("<br>Invalid Choice");

}

</script>

</body>

</html>

7)

<html>

<body>

<script type = "text/javascript">

var i=1;

while(i<=20)

{

document.write("<br>"+i);

i=i+2;

}

</script>

</body>

</html>

8)

<html>

<body>

<script type = "text/javascript">

var i;

for(i=1;i<=5;i++)

{

document.write("<br>"+i);

}

</script>

</body>

</html>

9)

<html>

<head>

<script type = "text/javascript">

function sayHello(name, age) {

document.write (name + " is " + age + " years old.");

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type = "button" onclick = "sayHello('John', 7)" value = "Say Hello">

</form>

<p>Use different parameters inside the function and then try...</p>

</body>

</html>

10) <html>

<body>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button"

onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>

Click Me!</button>

</body>

</html>

🟑🟑 🟑🟑🟑🟑🟑

**C programming language**  🡪 c is a compiler C is called by: - procedure oriented programming, procedure programming language or structural programming language.

Programming language is used to communicate with computers or electronics or electrical machine.

Program: = set of instruction given to computer to achieve a particular task.

Ex: = sum of two no’s. Soln: = step1:= read two numbers, step2:= let sum=a+b, step3:= print sum.

* **C**: = it is a modified version of BPCL. Basic programming combined language c was developed by Dennis Ritch at Bell laboratories of USA in 1972.
* **Character set of C: -** a-z, A-Z, 0-9 numbers.
* **Special characters** : - (). , : ; ‘ ‘ “ “ % # - \_ & ,&& !, ||, [ ], { } etc.
* **C tokens: -** smallest individual with of a program.
* **Keyword:** - reserved words used by c ex: = main, getch, print, scanf, int, float.
* **Identifies** : - used to identify definitions, variables, constants, structures etc ex: = int a; const int a;
* Struct student, sum ();
* **Variable** : - storage unit the value of which will change during the execution of the program ex:= coust float pi=3.142;
* **Data types:** - type of a data given to computer.

**A) primary data Type**: = (built in-data type we).

B) **Derived data type**: - derived by primary data type ex: = arrays, pointers, function.

C) **Uses defined data type**: = data type is defined by the user. Ex: = structures, class, unions, even etc.

1. **Primary data type**: = types: - int, float, string, char, getch. Int: = stores integer (short int 1 byte, int 2 bytes, and long int 4 bytes).
2. **Signed int**: = stores +ve & -ve values.
3. **Unsigned int**: = stores +ve values.
4. **Float**: = stores decimal or exponential numbers ex: = 3.142, 1.2345, 102 , 10+25, 10-25. Float-4 bytes double -8 bytes.
5. **chart**:= stores characters -char string, single characters—collection of characters, char c; char c[100];
6. **Operations:** = operator is a symbol to identify the operators ex:= + - add , - - sub =.
   1. **Arithmetic operations**: - add +, sub--, multiplication \*, division /, modulo division %, remainder is the answer 2)4, 4/0- remainder (2-quotient.
7. **Relational operators**: = used to relate or compare two variables constants. Operators: - ><, >=, <=, ==, !=, a>b, a<b, a==b.
8. **Logical operators**: = used to combine two express always logical operators work with relational operators. Operators: - and-&&, or-||, not-! Ex: = a>b && b>c && c>a. a==b && b==c && c==a. a==b || b==c || c==a.
9. **Conditional operators**: = used to check the condition operators:= : , ? . Syntax: - <exp1> : <exp2>? <exp3>--false (a>b): a? b.—true.
10. **Assignment operators**: = used to assign the value of right hand side of expression to left hand side. Operator:= -=, sum=a+b; increment operator:= increase the value of a variable by 1 , ++ , i=5 5, i++5,=6. Decrement operator: = decrease the value of variable by 1, - - , i=5 5, i- - 5=4. Expression: = combination of variable constant operator ex: = 4+5, x+5, x+y, 4+y. free processing statement.

* **struct of the program** :=

Documentation section

Link section

Global declaration

Main function

{

Local declaration

{

Function body

} Function 1; Function 2; \_ \_ \_ \_; \_ \_ \_ \_; }

# Documentation section

: =include the non-executable statements. This makes the program more readable.

**Comments**: = are used to write the statements. 🡪 Single line comments. //\_ \_ \_ \_ \_. 🡪 Multi line comments /\*\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \*/

* **Link section**: = we include header files in link section.
* **#include<stdio.h>** //stdio.h🡪 slandered input/output header files includes input/output functions.
* **#include<conio.h>** //conio.h🡪 console (screen) input/output header files includes input/output functions.
* **Printf ()**🡪 used to print the statement.
* **Scanf ()** 🡪 used to read the data.
* **Clrscr ()** 🡪 it clear the screen.
* **Getch ()**🡪 get the output on the screen.
* **Global declaration**: = the variables declared are used in all the functions of program ex: = int a, b;
* **Main function**: = execution starts from main.
* **Delimiters**: = {} 🡪indicates the beginning and ending the program.
* **Local declaration**: = variable declared will be used of accessed in that particular function.
* **Function body**: = actual code.

# 

# Programs: =

1. // program to print the message.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Printf (“welcome to c program”);

getch ();

}

1. // program to find sum and average of two numbers.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b;

Float sum, avg;

Clrscr ();

Printf (“enter the value of a and b :”);

Scanf (“%d%d”, &a, &b);

Sum=a+b;

Avg=sum/2;

Printf (“sum: =%f”, sum);

Printf (“\n average: =%f”, avg);

getch ();

}

1. // program to find simple interest.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int p, t;

Float r, si;

Clrscr ();

Printf (“enter the value of p, t and r :”);

Scanf (“%d%d%f”, &p, &t, &r);

Si= (p\*t\*r)/100;

Printf (“simple interest: =%f”, si);

getch ();

}

1. // program to find area and perimeter of square.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a;

Float area, peri;

Clrscr ();

Printf (“enter the value of a :”);

Scanf (“%d”, &a);

Area=a\*a;

Peri=4\*a;

Printf (“area: =%f”, area);

Printf (“\n perimeter: =%f”, peri);

getch ();

}

1. // program to find area and perimeter of rectangle.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int l, b;

Float area, peri;

Clrscr ();

Printf (“enter the value of l and b :”);

Scanf (“%d%d”, &l, &b);

Area=l\*b;

Peri=2\*(l+b);

Printf (“area: =%f”, area);

Printf (“\n perimeter: =%f”, peri);

getch ();

}

1. // program to find area and circumference of circle.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int r;

Float area, cir;

Clrscr ();

Printf (“enter the value of r :”);

Scanf (“%d”, &r);

Area=3.142\*r\*r;

Cir=2\*3.142\*r;

Printf (“area: =%f”, area);

Printf (“\n circumference: =%f”, cir);

getch ();

}

1. // program to find area of triangle.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int b, h;

Float area, triangle;

Clrscr ();

Printf (“enter the value of b and h :”);

Scanf (“%d%d”, &b, &h);

Area=b\*h;

Triangle=0.5\*b\*h;

Printf (“area: =%f”, area);

Printf (“\n triangle: =%f”, triangle);

getch ();

}

Soln: = b\*h=5\*5=25/5=0.5 area 1-2/2=1/2 0.5\*1/2=12.50 triangle.

1. // program to swap the values of two variables..

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b, temp;

Clrscr ();

Printf (“enter the value of a and b :”);

Scanf (“%d%d”, &a, &b);

Temp=a;

a=b;

b=temp;

Printf (“a after swap: =%f”, a);

Printf (“\n b after swap: =%f”, b);

getch ();

}

# Decision making and branching

Control statements:= used to transfer the control from one place to another.

# Unconditional control statements

Control statements:= used to transfer the control from one place to another in a program without conditional ex:= go to 🡪 use f7 function key.

Go to label;

\_ \_ \_ \_

\_ \_ \_ \_ forward jump

Label:

Label:

\_ \_ \_ \_ backward jump

\_ \_ \_ \_

Go to Label;

1. // program to demonstrate go to statement.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b;

Float sum, avg;

Char ch;

Label:

Clrscr ();

Printf (“enter the value of a and b :”);

Scanf (“%d%d”, &a, &b);

Sum=a+b;

Avg=sum/2;

Printf (“sum: =%f”, sum);

Printf (“\n average: =%f”, avg);

Printf (“\n do you want to continue sum? [y/n] :”);

Scanf (“%s”, &ch);

If (ch==’y’)

goto label;

getch ();

}

# Conditional operator

1. // program to find maximum of two no’s using conditional operator.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b, max;

Clrscr ();

Printf (“enter the value of a and b :”);

Scanf (“%d%d”, &a, &b);

Max=(a>b) ? a: b;

Printf (“%d= maximum of two no’s”, max);

getch ();

}

1. // program to find maximum of three no’s using conditional operator.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b, c, max1, max2;

Clrscr ();

Printf (“enter the value of a, b and c :”);

Scanf (“%d%d%d”, &a, &b, &c);

Max1= (a>b)? a: b;

Max2= (max1>c)? max1: c;

Printf (“%d= maximum of two no’s”, max2);

getch ();

}

# Mathematical function

1. // program to demonstrate mathematical function.

#include<stdio.h>

#include<conio.h>

#include<math.h>

Void main ()

{

Int x, y, z;

Float a, b, c;

Clrscr ();

Printf (“enter the number :”);

Scanf (“%d”, &x);

Printf (“enter the power :”);

Scanf (“%d”, &y);

Printf (“enter the –ve number :”);

Scanf (“%d”, &z);

A=sqrt(x);

B=POW(x, y);

C=abs (z);

Printf (“\n square root of a number: %f”, a);

Printf (“\n power of a number: %f”, b);

Printf (“\n absolute of a number: %f”, c);

getch ();

}

1. // program to find compound interest.

#include<stdio.h>

#include<conio.h>

#include<math.h>

Void main ()

{

Long Int p;

Int n;

Float r, ci, a;

Clrscr ();

Printf (“enter the value of p, n and r :”);

Scanf (“%d%d%f”, &p, &n, &r);

a= p\*((1+r/100),n);

ci=a-p;

Printf (“amount: =%f”, a);

Printf (“\n compound interest : =%f”, ci);

getch ();

}

# Conditional control statements

Control transfers from one place to another place in a program according to condition.

Ex: = if, switch. a) Simple if, b) if \_ \_ \_ \_ else. C) If \_ \_ \_ \_ else if ladder, d) nested if.

1. **Simple if**: = checks the condition if true statement executes otherwise goes to next statement.

Syntax: = if <condition>

{

Statement

}

Statement 🡪x;

1. // program to demonstrate simple if statement.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int money;

Clrscr ();

Printf (“enter the money :”);

Scanf (“%d”, &money);

If(money==0)

Printf (“borrow money”);

getch ();

}

1. // program to find even or odd no’s.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a;

Clrscr ();

Printf (“enter the value of a :”);

Scanf (“%d”, &a);

If (a%2==0)

Printf (“%d is even”, a);

If (a%2! =0)

Printf (“%d is odd”, a);

getch ();

}

1. **If\_ \_ \_ \_else statements**: = if condition is true executes of block statements otherwise execute false block statement two-way branching statement.

Syntax: = if <condition>

{True--statement

Else

} False--statement

Statement 🡪x;

1. // program to find biggest of two no’s.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b;

Clrscr ();

Printf (“enter the value of a and b :”);

Scanf (“%d%d”, &a, &b);

If (a>b)

Printf (“%d is big”, a);

Else

Printf (“%d is big”, b);

getch ();

}

1. // program to find +ve or -ve no’s.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a;

Clrscr ();

Printf (“enter the value of a :”);

Scanf (“%d”, &a);

If (a>0)

Printf (“%d is +ve”, a);

Else

Printf (“%d is -ve”, a);

getch ();

}

1. **Nested if**: = checks the condition if true checks next condition if true checks next & so on.

Syntax: = if <condition1>

{True—statement1;

If <condition2>

{True—statement2;

Else

False—statement1;

} else

False—statement1;

1. // program to find biggest of three no’s.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b, c;

Clrscr ();

Printf (“enter the value of a, b and c :”);

Scanf (“%d%d%d”, &a, &b, &c);

If (a>b)

{

If (a>b)

Printf (“%d is big”, a);

Else

Printf (“%d is big”, c);

}

Else

{

If (b>c)

Printf (“%d is big”, b);

Else

Printf (“%d is big”, c);

}

getch ();

}

1. // program to find the eligible voting or not.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int age, n;

Clrscr ();

Printf (“enter the age :”);

Scanf (“%d”, &age);

Printf (“enter the nationality as 1 if Indian :”);

Scanf (“%d”, &n);

If (age>=18)

{

If (n==1)

Printf (“Eligible for voting”);

Else

Printf (“Not Eligible for voting”);

}

Else

Printf (“Not Eligible for voting”);

getch ();

}

1. **If\_ \_ \_ \_else if ladders:** = checks the condition if true executes statements, if false checks next condition & so \_\_ \_ \_ on.

Syntax: = if <condition1>

{True—statement1;

Else If <condition2>

{True—statement2;

Else if <condition3>

{True—statement3;

\_ \_ \_ \_

\_ \_ \_ \_

Else

D—statement1;

1. // program to find biggest of three no’s using it.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b, c;

Clrscr ();

Printf (“enter the value of a, b and c :”);

Scanf (“%d%d%d”, &a, &b, &c);

If (a>b && a>c)

Printf (“%d is big”, a);

Else If (b>c && c>a)

Printf (“%d is big”, b);

Else

Printf (“%d is big”, c);

getch ();

}

1. // program to checks whether the triangle is equilateral, isosceles or scalene.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b, c;

Clrscr ();

Printf (“enter the value of a, b and c :”);

Scanf (“%d%d%d”, &a, &b, &c);

If (a==b && b==c && c==a)

Printf (“triangle is equilateral”);

Else If (a==b || b==c || c==a)

Printf (“triangle is isosceles”);

Else

Printf (“triangle is scalene”);

getch ();

}

1. // program to find the student marks detail.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int rollno, STD, total, kan, eng, hin;

Float per;

Char name [10];

Clrscr ();

Printf (“enter the rollno :”);

Scanf (“%d”, &rollno);

Printf (“enter the student Name :”);

Scanf (“%s”, &name);

Printf (“enter the standard :”);

Scanf (“%d”, &STD);

Printf (“enter the marks of kannada :”);

Scanf (“%d”, &Kan);

Printf (“enter the marks of English :”);

Scanf (“%d”, &eng);

Printf (“enter the marks of Hindi :”);

Scanf (“%d”, &hin);

Total=Kan+eng+hin;

Per=total/3;

Printf (“\n\*\*\*\*student detail\*\*\*\*” “);

Printf (“\n student rollno: =%d”, rollno);

Printf (“\n student standard: =%d”, STD);

Printf (“\n student name: =%s”, name);

Printf (“\n student total: =%d”, total);

Printf (“\n student percentage: =%f”, per);

Printf (“\n student Grade: =”);

{

If (per>=75)

Printf (“A”);

Else If (per>=75 && per<=60)

Printf (“B”);

Else

Printf (“C”);

Printf (“\n\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*”);

}

getch ();

}

1. // program to find the Grade of the Employee

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int code;

Float basic, hra, da, pf, lic, gross, ded, ns;

Char name [10], design;

Clrscr ();

Printf (“enter the code :”);

Scanf (“%d”, &code);

Printf (“enter the employee Name :”);

Scanf (“%s”, &name);

Printf (“enter the Designation :”);

Scanf (“%s”, &design);

Printf (“enter the basic salary :”);

Scanf (“%f”, &basic);

Hra=basic\*0.5;

Da=basic\*0.3;

Pf=basic\*0.1;

Lic=500;

Gross=Basic+Hra+Da;

Ded=pf+Lic;

Ns=Gross-Ded;

Printf (“\n\*\*\*\*Employee Detail\*\*\*\*” “);

Printf (“\n Employee Code: =%d”, code);

Printf (“\n Designation: =%s”, design);

Printf (“\n Employee name: =%s”, name);

Printf (“\n Employee Net Salary: =%f”, ns);

Printf (“\n Employee Grade: =”);

{

If (ns>=25000)

Printf (“A”);

Else If (ns>=25000 && per<=20000)

Printf (“B”);

Else If (ns>=20000 && per<=10000)

Printf (“C”);

Else

Printf (“D”);

Printf (“\n\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*”);

}

getch ();

}

1. // program to find the commission of the sales.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int code, qty;

Float amount, comm, rate;

Char product [10];

Clrscr ();

Printf (“enter the code :”);

Scanf (“%d”, &code);

Printf (“enter the product :”);

Scanf (“%s”, &product);

Printf (“enter the quantity :”);

Scanf (“%d”, &qty);

Printf (“enter the Rate of product :”);

Scanf (“%f”, &rate);

Amount=qty\*rate;

Comm=amount\*0.2;

Printf (“\n\*\*\*\*Commission of Sales\*\*\*\*” “);

Printf (“\n product Code: =%d”, code);

Printf (“\n Product: =%s”, product);

Printf (“\n Quantity: =%d”, qty);

Printf (“\n Total Sales Amount: =%f”, amount);

Printf (“\n Commission: =%f”, comm);

Printf (“\n\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*”);

getch ();

}

1. // program to find the Units of Electric city bill.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int rrno, units, units1, units2;

Float amount;

Char name [10];

Clrscr ();

Printf (“enter the rrno :”);

Scanf (“%d”, &rrno);

Printf (“enter the Name :”);

Scanf (“%s”, &name);

Printf (“enter the reading units :”);

Scanf (“%d”, &units);

If (units<=100)

Amount=units\*2.5;

Else if ((units<=200 && units>=100)

Amount= (100\*2.5) + ((units-100)\*4);

Else

{

Units1=units-100;

Units2=units-100;

Amount= (100\*2.5)+units1\*4+units2\*5;

}

Printf (“\n\*\*\*\*Electric city Bill Detail\*\*\*\*” “);

Printf (“\n meter reading number: =%d”, rrno);

Printf (“\n name: =%s”, name);

Printf (“\n units: =%d”, units);

Printf (“\n amount: =%f”, amount);

Printf (“\n\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*”);

}

getch ();

}

1. // program to find the Telephone bill.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int phno, calls, callrate;

Float amount;

Char name [10];

Clrscr ();

Printf (“enter the phone number :”);

Scanf (“%d”, &phno);

Printf (“enter the Name :”);

Scanf (“%s”, &name);

Printf (“enter the calls :”);

Scanf (“%d”, &calls);

{

If (calls<=100)

Callrate=calls\*0.40;

Else If (calls>=100)

Callrate=calls\*0.60;

Else If (calls<=200 && calls>=100)

Callrate=calls\*0.80;

Else

Callrate=calls\*0.20;

Amount=calls\*callrate;

Printf (“\n\*\*\*\*Electric city Bill Detail\*\*\*\*” “);

Printf (“\n phone number: =%d”, phno);

Printf (“\n name: =%s”, name);

Printf (“\n calls: =%d”, calls);

Printf (“\n call Rate: =%d”, callrate);

Printf (“\n amount: =%f”, amount);

Printf (“\n\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*”);

}

getch ();

}

1. // program to find the Bank Account detail.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int accno;

Float amount, Bal, interest;

Char name [10],account;

Clrscr ();

Printf (“enter the account number :”);Scanf (“%d”, &accno);

Printf (“enter the Name :”);

Scanf (“%s”, &name);

Printf (“enter the balance :”);

Scanf (“%f”, &bal);

Printf (“enter the account type :”);

Scanf (“%s”, &account);

Printf (“\n\*\*\*\*Account Detail\*\*\*\*” “);

Printf (“\n account number: =%d”, accno);

Printf (“\n name: =%s”, name);

Printf (“\n account: =%s”, account);

If (account==’s’)

{

Interest=0.05\*bal;

Printf (“\n S.B Account”);

}

Else If (account==’r’)

{

Interest=0.06\*bal;

Printf (“\n Recurring Deposit”);

}

Else

Interest=0.09\*bal;

Printf (“\n Fixed Deposit”);

Amount=bal+ interest;

Printf (“\n amount: =%f”, amount);

Printf (“\n\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*”);

getch ();

}

# Switch Case

1. // program to demonstrate switch case.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Char code;

Clrscr ();

Printf (“enter the code :”);

Scanf (“%s”, &code);

Switch (code)

{

Case ‘b’:

Printf (“boy”);

Break;

Case ‘g’:

Printf (“girl”);

Break;

Default:

Printf (“invalid code”);

}

getch ();

}

1. // program to demonstrate switch case.

#include<stdio.h>

#include<conio.h>

#include<math.h>

Void main ()

{

Int a, b;

Float sum, diff, prod, quo, sqrt, mod;

Char op;

Clrscr ();

Printf (“enter the value of a and b :”);

Scanf (“%d%d”, &a, &b);

Printf (“enter the operator :”);

Scanf (“%s”, &op);

Switch (op)

{

Case ‘+’:

Sum=a+b;

Printf (“sum=%f”, sum);

Break;

Case ‘-’:

Diff=a-b;

Printf (“difference=%f”, diff);

Break;

Case ‘\*’:

Prod=a\*b;

Printf (“product=%f”, prod);

Break;

Case ‘/’:

Quo=a/b;

Printf (“quotient=%f”, quo);

Break;

Case ‘^’:

Sqrt=sqrt (a);

Printf (“square root of a=%f”, sqrt);

Break;

Case ‘%’:

Mod=a%b;

Printf (“Reminder =%f”, mod);

Default:

Printf (“invalid operator”);

}

getch ();

}

1. // program to find the areas of geometrical figure using switch case..

#include<stdio.h>

#include<conio.h>

Void main ()

{

Float a, a1, a2, a3, a4, b, h, l, w, r1;

Char code;

Clrscr ();

Printf (“enter the code :”);

Scanf (“%s”, &code);

Switch (code)

{

Case ‘r’:

Printf (“enter the value of l and w:”);

Scanf (“%f%f”, &l, &w);

A1=l\*w;

Printf (“area=%f”, a1);

Break;

Case ‘s’:

Printf (“enter the value of a :”);

Scanf (“%f”, &a);

A2=a\*a;;

Printf (“area=%f”, a2);

Break;

Case ‘+’:

Printf (“enter the value of b and h :”);

Scanf (“%f%f”, &b, &h);

A3=0.5\*b\*h;

Printf (“area=%f”, a3);

Break;

Case ‘c’:

Printf (“enter the value of r1 :”);

Scanf (“%f”, &r1);

A4=3.142\*r1\*r1;

Printf (“area=%f”, a4);

Break;

Default:

Printf (“invalid code”);

}

getch ();

}

# Looping

**Increment and decrement operator using while, do + while, for methods.**

1. // program to display 10 natural numbers using do while.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=1;

Clrscr ();

do

{

Printf (“%d\n”, i);

I++;

} while (i<=10);

getch ();

}

1. // program to display 10 natural numbers using while.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=1;

Clrscr ();

While (i<=10)

{

Printf (“%d\n”, i);

I++;

}

getch ();

}

1. // program to display 10 natural numbers using For.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i;

Clrscr ();

For (i=1; i<=10; i++)

Printf (“%d\n”, i);

getch ();

}

1. // program to display 10 natural numbers in reverse using do while.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=10;

Clrscr ();

do

{

Printf (“%d\n”, i);

I--;

} while (i>=1);

getch ();

}

1. // program to display 10 natural numbers using while.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=10;

Clrscr ();

While (i>=1)

{

Printf (“%d\n”, i);

I--;

}

getch ();

}

1. // program to display 10 natural numbers using For.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i;

Clrscr ();

For (i=10; i>=1; i--)

Printf (“%d\n”, i);

getch ();

}

1. // program to display n natural numbers using do while.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=1 , n;

Clrscr ();

Printf (“enter the value of n:”);

Scanf (“%d”, &n);

do

{

Printf (“%d\n”, i);

I++;

} while (i<=10);

getch ();

}

1. // program to display n natural numbers using while.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=1, n;

Clrscr ();

Printf (“enter the value of n:”);

Scanf (“%d”, &n);

While (i<=10)

{

Printf (“%d\n”, i);

I++;

}

getch ();

}

1. // program to display n natural numbers using For.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=1, n;

Clrscr ();

Printf (“enter the value of n:”);

Scanf (“%d”, &n);

For (i=1; i<=10; i++)

Printf (“%d\n”, i);

getch ();

}

1. // program to display sum of n natural numbers using do while.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=1, n, sum=0;

Clrscr ();

Printf (“enter the value of n:”);

Scanf (“%d”, &n);

do

{

Printf (“%d\n”, i);

Sum=sum+i;

I++;

} while (i<=10);

Printf (“sum=%d”, sum);

getch ();

}

1. // program to display sum of n natural numbers using while.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=1, n, sum=0;

Clrscr ();

Printf (“enter the value of n :”);

Scanf (“%d”, &n);

While (i<=10)

{

Printf (“%d\n”, i);

Sum=sum+i;

I++;

}

Printf (“sum=%d”, sum);

getch ();

}

1. // program to display sum of n natural numbers using For.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i=1, n, sum=0;

Clrscr ();

Printf (“enter the value of n :”);

Scanf (“%d”, &n);

For (i=1; i<=10; i++)

Printf (“%d\n”, i);

Sum=sum+i;

Printf (“sum=%d”, sum);

getch ();

}

1. // program to find the factorial of a number.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int i, n, fact=1;

Clrscr ();

Printf (“enter the value of n :”);

Scanf (“%d”, &n);

For (i=1; i<=n; i++)

{

Fact=fact\*I;

}

Printf (“factorial of a number=%d”, fact);

getch ();

}

1. // program to find the LCM & GCD of 2 number.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a, b, m, n, LCM, gcd;

Clrscr ();

Printf (“enter the value of a and b :”);

Scanf (“%d%d”, &a, &b);

M=a;

N=b;

While (a!=b)

{

If (a>b)

A=a-b;

Else

B=b-a;

}

LCM= (m\*n)/a;

Printf (“GCD=%d”, gcd);

Printf (“LCM=%d”, LCM);

getch ();

}

1. // program to generate to print first n Fibonacci number.

#include<stdio.h>

#include<conio.h>

#include<math.h>

Void main ()

{

Int fst=0, snd=1, n, count=2, next;

Clrscr ();

Printf (“enter the value of n :”);

Scanf (“%d”, &n);

Printf (“%d\n%d\n”, fst, snd);

While (count<n)

{

Next=fst+ snd;

Printf (“%d\n”, next);

Count++;

Fst=snd;

Snd=next;

}

getch ();

}

# Arrays

**Arrays**: = collection of the data items which share a common name & data type. Ex: = int a [5]; int🡪 type, a[5]🡪 array name, [5]🡪 index (size).

Single dimensional arrays: = store a list of values.

Syntax: = type array name [index]; ex: = int a [5];

Initialization: = int a [5] = {1, 2, 3, 4, 5};

Double dimensional arrays: = store a value in rows & columns (table)

Syntax: = types of array name [row-size] [column] ex: = int a [3] [3];

Initialization: = int a [2] [2] = {1, 2, 3, 4};

1. // program to demonstrate single dimensional array.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a [5], i;

Clrscr ();

Printf (“enter the elements of array :”);

For (i=1; i<=5; i++)

Scanf (“%d”, &a[i]);

Printf (“elements of array are\n”);

For (i=1; i<=5; i++)

Printf (“%d”, a[i]);

getch ();

}

1. // program to find sum & avg of array elements.

#include<stdio.h>

#include<conio.h>

Void main ()

{

Int a [100], I, n, sum=0, avg=0;

Clrscr ();

Printf (“enter the value of n :”);

Scanf (“%d”, &n);

Printf (“enter the elements of array :”);

For (i=1; i<=n; i++)

Scanf (“%d”, &a[i]);

Printf (“elements of array are\n”);

For (i=1; i<=5; i++)

{

Sum=sum + a [i];

}

Avg= sum/n;

Printf (“\n sum=%d”, sum);

Printf (“\n average=%d”, avg);

getch ();

}

**What is C++:**

C++ is an object-oriented programming language. It is an extension to C programming. C++ programming language was developed in 1980 by **BjarneStroustrup** at bell laboratories of AT&T (American Telephone & Telegraph), located in U.S.A. C++ runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX..

**BjarneStroustrup** is known as the **founder of C++ language.**

**Object-Oriented Programming (OOPs)**

C++ supports the object-oriented programming, the four major pillar of object oriented programming used in C++ are:

**1. Inheritance 2.Polymorphism 3.Encapsulation 4.Abstraction**

**Usage of C++**

By the help of C++ programming language, we can develop different types of secured and robust applications:

* Window application
* Client-Server application
* Device drivers
* Embedded firmware etc.

**C vs C++**

|  |  |  |
| --- | --- | --- |
| **No.** | **C** | **C++** |
| 1) | C follows the **procedural style programming.** | C++ is multi-paradigm. It supports both **procedural and object oriented.** |
| 2) | Data is less secured in C. | In C++, you can use modifiers for class members to make it inaccessible for outside users. |
| 3) | C follows the **top-down approach.** | C++ follows the **bottom-up approach.** |
| 4) | C does not support function overloading, Operator Overloading etc. OOP concepts. | C++ supports function overloading, Operator Overloading etc. OOP concepts.. |
| 5) | In C, you can't use functions in structure. | In C++, you can use functions in structure. |
| 6) | In C, **scanf() and printf()** are mainly used for input/output. | C++ mainly uses stream **cin and cout** to perform input and output operations. |

**C++ Features**:

**C++ is object oriented programming language. It provides a lot of features that are given below.**

* The main focus remains on data rather than procedures.
* Object-oriented programs are segmented into parts called objects.
* Data structures are designed to categorize the objects.
* Data member and functions are tied together as a data structure.
* Data can be hidden and cannot be accessed by external functions using access specifier .
* Objects can communicate among themselves using functions.
* New data and functions can be easily added anywhere within a program whenever required.
* Since this is an object-oriented programming language, it follows a bottom up approach, i.e. the execution of codes starts from the main which resides at the lower section and then based on the member function call the working is done from the classes.

**C++ Program:** Basically, a C++ program involves the following section:

Documentation

Pre-processor Statements

Global Declarations

The main () function{

Local Declarations

Program Statements & Expressions}

User Defined Functions

**Example : //program to display a line of text.**

#include<iostream.h>

#include<conio.h>

void main()

{

clrscr();

cout<<”welcome to c++ programming language”;

getch();

}

**Standard output stream (cout):**The **cout** is a predefined object of **ostream** class. It is connected with the standard output device, which is usually a display screen. The cout is used in conjunction with stream **insertion operator** **(<<)** to display the output on a console

**Standard input stream (cin):**The **cin** is a predefined object of **istream** class. It is connected with the standard input device, which is usually a keyboard. The cin is used in conjunction with stream **extraction operator (>>)** to read the input from a console.

**Standard end line (endl):**The **endl** is a predefined object of **ostream** class. It is used to insert a new line characters and flushes the stream.

**C++ Variable:** A variable is a name of memory location. It is used to store data. Its value can be changed and it can be reused many times.

**Data\_type  variable list;**

The example of declaring variable is given below: **1) int x; 2) float y; 3) char z;**

Here, x, y, z are variables and int, float, char are data types.

We can also provide values while declaring the variables as given below:

**1) int x=5,b=10;  //declaring 2 variable of integer type    2)float f=30.8;    3)char c='A';**

* A variable can have alphabets, digits and underscore.
* A variable name can start with alphabet and underscore only. It can't start with digit.
* No white space is allowed within variable name.
* A variable name must not be any reserved word or keyword e.g. chars, float etc.

**C++ Data Types : There are 4 types of data types in C++ language.**

|  |  |
| --- | --- |
| **Types** | **Data Types** |
| Basic Data Type | int, char, float, double, etc |
| Derived Data Type | array, pointer, etc |
| User Defined Data Type | structure |

C++ Keywords

A keyword is a reserved word. You cannot use it as a variable name, constant name etc.

Example: if, else,switch,case,do,while,for,class etc

**C++ Operators:** Operators are special type of functions that takes one or more arguments and produces a new value.

* Arithmetic Operators [+ ,- ,\* ,/ ,%]
* Relational Operators ]> ,>= ,< ,<= ,== ,!=]
* Logical Operators [&& , ||, ! ]
* Assignment Operator [=]
* Unary operator [++ ,--]
* Ternary or Conditional Operator [ ? , : ]
* sizeof operator [sizeof(nt)]
* pointer to member operator [ .\* and ->\*]
* **scope resolution** operator [ :: ]
* **address** operator [ & ]
* **new o**perator (memory allocation operator)
* **delete** operator(memory release operator)

**C++ Control Statement**

**if-else**

In C++ programming, if statement is used to test the condition. There are various types of if statements in C++.

* **if statement**
* **if-else statement**
* **if-else-if ladder**

**IF Statement**: The C++ if statement tests the condition. It is executed if condition is true.

**Syntax: if(condition) {**

**//code to be executed }**

Example: #include <iostream.h>

#include<conio.h>

void main ()

{

int num = 10;

if (num % 2 == 0) { cout<<"It is even number"; }

getch();

}

**if-else Statement:** The C++ if-else statement also tests the condition. It executes if block if condition is true otherwise else block is executed.

Syntax: if(condition) { //code if condition is true }

else { //code if condition is false }

Example:

#include <iostream.h>

#include<conio.h>

void main () {

int num;

cout<<"Enter a Number: ";

cin>>num;

if (num % 2 == 0) { cout<<"It is even number"<<endl; }

else { cout<<"It is odd number"<<endl; }

getch();

}

**if-else-if ladder :** statement executes one condition from multiple statements.

**Syntax:**

if(condition1){ //code to be executed if condition1 is true }

else if(condition2){ //code to be executed if condition2 is true }

else if(condition3){ //code to be executed if condition3 is true }

...

else{//code to be executed if all the conditions are false }

Example:

#include<iostream.h>

#include<conio.h>

void main()

{

int avg;

clrscr();

cout<<"Enter average marks obtained in 5 subjects :";

cin>>avg;

cout<<"Your Grade is ";

if(avg>80)

{ cout<<"A";}

else if(avg>60 && avg<=80)

{ cout<<"B";}

else if(avg>40 && avg<=60)

{ cout<<"C";}

else

{ cout<<"D";}

getch();

}

**switch:** The C++ switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement in C++.

**Syntax:**  switch (expression)

{

case label1: statements ;

break;

case label2: statements;

break;

……

default: staements;

break;

}

**Example**

**// Program to build a simple calculator using switch Statement**

#include <iostream.h>

#include<conio.h>

void main()

{

char op;

float num1, num2;

clrscr();

cout << "Enter an operator (+, -, \*, /): ";

cin >> op;

cout << "Enter two operands: ";

cin >> num1 >> num2;

switch (op)

{

case '+':

cout << num1 << " + " << num2 << " = " << num1+num2;

break;

case '-':

cout << num1 << " - " << num2 << " = " << num1-num2;

break;

case '\*':

cout << num1 << " \* " << num2 << " = " << num1\*num2;

break;

case '/':

cout << num1 << " / " << num2 << " = " << num1/num2;

break;

default:

// operator is doesn't match any case constant (+, -, \*, /)

cout << "Error! operator is not correct";

break;

}

getch();

}

**Looping**:

Loops are used in programming to repeat a specific block until some end condition is met. There are three types of loops in C++ programming:

1. for loop
2. while loop
3. do while loop

**While loop:**

**Syntax:** while**(condition){**

**//code to be executed  }**

**Example**:

// C++ Program to compute factorial of a number

// Factorial of n = 1\*2\*3...\*n

#include <iostream.h>

#include<conio.h>

void main()

{

int number, i = 1, factorial = 1;

clrscr();

cout << "Enter a positive integer: ";

cin >> number;

while ( i <= number)

{

factorial \*= i; //factorial = factorial \* i;

++i;

}

cout<<"Factorial of "<< number <<" = "<< factorial;

getch();

}

**Do-While Loop** :

**Syntax:** do

{ //code to be executed

}while(condition);

**Example** : //C++ Program to display natural numbers up to 10.

#include <iostream>

#include<conio.h>

void  main()

 {

     int i = 1;

          do{

              cout<<i<<"\n";

              i++;

          } while (i <= 10) ;

getch();

}

**Example:**

**Syntax:** for(initialization; condition; incr/decr) {

//code to be executed    }

**Example: //program to display your name for specified number of time**

#include <iostream.h>

#include<conio.h>

void main()

{

int time,i;

clrscr();

cout << "how many times you want to display your name";

cin >>time;

for(i=0;i<time;i++)

{

0 Cout<<”\n KEONICS”;

}

getch();

}

**C++ Functions**

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is main().

**Advantages of using functions in a program**

* You can divide your program in logical blocks. ...
* Use of function avoids typing same pieces of code multiple times. ...
* Individual functions can be easily tested.
* In case of any modification in the code you can modify only the function without changing the structure of the program.

Types of Functions

There are two types of functions in C programming:

**1. Library Functions:** are the functions which are declared in the C++ header files such as ceil(x), floor(x), sqrt(x),sin(x) etc.

**2. User-defined functions:** are the functions which are created by the C++ programmer, so that he/she can use it many times. It reduces complexity of a big program and optimizes the code.

**Declaration of a function:**

**Syntax:**

**return\_type function\_name( parameter list ) {**

**body of the function}**

**Example:**

#include<iostream.h>

#include<conio.h>

// function declaration

int max(int num1, int num2);//function Declaration

void main () {

// local variable declaration:

int a = 100;

int b = 200;

int ret;

// calling a function to get max value.

ret = max(a, b);

cout << "Max value is : " << ret << endl;

getch();

}

// function returning the max between two numbers

int max(int num1, int num2) {

// local variable declaration

int result;

if (num1 > num2)

result = num1;

else

result = num2;

return (result);

}

**Inline functions**

When a function is declared as inline, the compiler places a copy of the code of that specific function at each point where the function is called at compile time.

**Syntax:**

inline return\_type function\_name(arguments...)

{

//function\_code

//return\_value;

}

**Example:**

//Simple Inline Function Example Program in C++

#include<iostream.h>

#include<conio.h>

inline int square(int x);

void main() {

int a = 100, b = 200;

cout << "Simple Inline Function Example Program in C++\n";

cout << "\nSquare value for " << a << " is :" << square(a);

cout << "\nSquare value for " << b << " is :" << square(b);

getch();

}

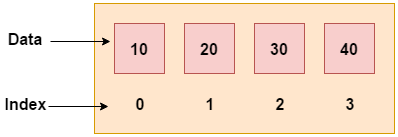
// Inline square function

inline int square(int x) {

return (x \* x);}

**C++ Arrays**

Array is a group of similar types of elements that have contiguous memory location.

In C++, array index starts from **0 to (n-1)** 

***Advantages of C++ Array***

* Code Optimization (less code)
* Random Access
* Easy to traverse data
* Easy to manipulate data
* Easy to sort data etc.

***Disadvantages of C++ Array ->*** Fixed size

***Array Types :***

There are 2 types of arrays in C++ programming:

1. Single Dimensional Array
2. Multidimensional Array

***Single Dimensional Array:***

A one-dimensional array is a group of elements having the same data type and same name.

**Declaration: data\_type array\_name[array\_size];**

**Initialization: data\_type array\_name[array\_size]=comma\_separated\_element\_list};**

**Example :** // C++ One Dimensional Array

#include<iostream.h>

#include<conio.h>

void main()

{

clrscr();

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

int i;

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

{ cout<<"arr["<<i<<"] = "<<arr[i]<<"\n";}

getch();

}

**C++ Passing Array to Function**

**functionname(arrayname); //passing array to function**

**Example:**

#include <iostream.h>

#include<conio.h>

void printArray(int arr[5]);

void main()

{

        int arr1[5] = { 10, 20, 30, 40, 50 };

        int arr2[5] = { 5, 15, 25, 35, 45 };

        printArray(arr1); //passing array to function

        printArray(arr2);

}

void printArray(int arr[5])

{

    cout << "Printing array elements:"<< endl;

    for (int i = 0; i < 5; i++)

    {

                   cout<<arr[i]<<"\n";

    }

getch();

}

C++ Multidimensional Arrays:

**Declare Two Dimensional Array->** data\_type array\_name[row\_size][column\_size];

**Initialization:**

data\_type array\_name[row\_size][column\_size] = { comma\_separated\_value\_list} };

**Example :**

// C++ Two Dimensional Array

#include<iostream.h>

#include<conio.h>

void main()

{

clrscr();

int arr[5][2] = { {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6} };

int i, j;

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

{

for(j=0; j<2; j++)

{

cout<<"arr["<<i<<"]["<<j<<"] = "<<arr[i][j]<<"\n";

}

}

getch();

}

**C++ OOPs Concepts**

The major purpose of C++ programming is to introduce the concept of object orientation to the C programming language. Object Oriented Programming is a paradigm that provides many concepts such as **inheritance, data binding, polymorphism etc.**

The programming paradigm where everything is represented as an object is known as truly object-oriented programming language.

**OOPs (Object Oriented Programming System):**

**Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

**Object**

Any entity that has state and behaviour is known as an object.

For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

Class

**Collection of objects** is called class. It is a logical entity.

**Inheritance**

**When one object acquires all the properties and behaviours of parent object** i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**Polymorphism**

When **one task is performed by different ways** i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.In C++, we use Function overloading and Function overriding to achieve polymorphism.

**Abstraction**

**Hiding internal details and showing functionality** is known as abstraction. For example: phone call, we don't know the internal processing.In C++, we use abstract class and interface to achieve abstraction.

**Encapsulation**

**Binding (or wrapping) code and data together into a single unit is known as encapsulation.**

For example: capsule, it is wrapped with different medicines.

***Advantage of OOPs over Procedure-oriented programming language***

* OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.
* OOPs provide data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.
* OOPs provide ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

**C++ Object and Class**

Since C++ is an object-oriented language, program is designed using objects and classes in C++.

**C++ Object:**

Object is a runtime entity, it is created at runtime. Object is an instance of a class. All the members of the class can be accessed through object.

**C++ Class**

In C++, object is a group of similar objects.

**Syntax to create class:**

**Class class\_name**

**{**

**Data Members;**

**Methods;**

**}**

**Where class-|>keyword**

**Class Name->any valid identifies**

**Example:** class A

{

public:

double length; // Length of a box

double breadth; // Breadth of a box

double height; // Height of a box

}

* **Private**, **Protected**, **Public** is called visibility labels.
* The members that are declared private can be accessed only from within the class.
* Public members can be accessed from outside the class also.
* In C++, data can be hidden by making it private.

**C++ Object and Class Example**

Let's see an example of class that has two fields: id and name. It creates instance of the class, initializes the object and prints the object value.

#include <iostream.h>

#include <conio.h>

class Student {

   public:

      int id;//data member (also instance variable)

      string name;//data member(also instance variable)

};

void main() {

    Student s1; //creating an object of Student

    s1.id = 201;

    s1.name = "Sonoo Jaiswal";

    cout<<s1.id<<endl;

    cout<<s1.name<<endl;

    getch();

}

**C++ Class Example: Store and Display Employee Information**

#include <iostream.h>

#include <conio.h>

class Employee {

   public:

       int id;//data member (also instance variable)

       float salary; //data member(also instance variable)

       void insert(int i, float s)

        {

            id = i;

salary = s;

        }

       void display()

        {

            cout<<”id=”<<id<<"\nsalary= "<<salary<<endl;

        }

};

void main() {

    Employee e1; //creating an object of Employee

    Employee e2; //creating an object of Employee

    e1.insert(201,15000);

    e2.insert(202, 29000);

    e1.display();

    e2.display();

    getch(); }

**C++ Constructor**

In C++, constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally. The constructor in C++ has the same name as class or structure.

**There can be two types of constructors in C++.**

1. Default constructor
2. Parameterized constructor

**C++ Default Constructor** :A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.

Example:

#include <iostream.h>

#include <conio.h>

class Employee

{

public:

Employee()

{

cout<<"Default Constructor Invoked"<<endl;

}

};

void main()

{

Employee e1; //creating an object of Employee

Employee e2;

getch(); }

**C++ Destructor**

A destructor works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically.

A destructor is defined like constructor. It must have same name as class. But it is prefixed with a **tilde sign (~).**

**C++ friend function**

If a function is defined as a friend function in C++ then the protected and private data of a class can be accessed using the function. By using the keyword **friend** compiler knows the given function is a friend function .For accessing the data, the declaration of a friend function should be done inside the body of a class starting with the keyword **friend.**

**Declaration of friend function in C++**

**Syntax:**

class class\_name

{

    friend data\_type function\_name(arguments);

};

**Example:** #include<iostream.h>

#include<conio.h>

class base {

int val1, val2;

public:

void get() {

cout << "Enter two values:";

cin >> val1>>val2;

}

friend float mean(base ob);

};

float mean(base ob) {

return float(ob.val1 + ob.val2) / 2;

}

void main() {

clrscr();

base obj;

obj.get();

cout << "\n Mean value is : " << mean(obj);

getch(); }

**C++ Inheritance**

In C++, inheritance is a process in which one object acquires all the properties and behaviours of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviours which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

**Forms of inheritance in c++:**

|  |  |
| --- | --- |
| Single inheritance | When one class inherits another class, it is known as single level inheritance |
| Multiple inheritance | One derived class with several base classes is called multiple inheritance |
| Hierarchical inheritance | One class inherits more than one derived class is known as hierarchical inheritance |
| Multilevel inheritance | When one class inherits another class which is further inherited by another class, it is known as multi-level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes. |
| Hybrid inheritance | Combination of all forms of inheritance is known as hybrid inheritance. |

**Creation of derived class:**

**Syntax:**

class derived-class: visibility-mode base-class

{

Class body

}

**C++ Single Inheritance Example:**

#include<iostream.h>

#include<conio.h>

class emp {

public:

int eno;

char name[20], des[20];

void get() {

cout << "Enter the employee number:";

cin>>eno;

cout << "Enter the employee name:";

cin>>name;

cout << "Enter the designation:";

cin>>des;

}

};

class salary : public emp

{

float bp, hra, da, pf, np;

public:

void get1() {

cout << "Enter the basic pay:";

cin>>bp;

cout << "Enter the HRA:";

cin>>hra;

cout << "Enter the DA :";

cin>>da;

cout << "Enter the PF";

cin>>pf;

}

void calculate() { np = bp + hra + da - pf; }

void display() {

cout << eno << "\t" << name << "\t" << des << "\t" << bp << "\t" << hra << "\t" << da << "\t" << pf << "\t" << np << "\n";

}

};

void main()

{

int i, n;

char ch;

salary s[10];

clrscr();

cout << "Enter the number of employee:";

cin>>n;

for (i = 0; i < n; i++)

{ s[i].get();

s[i].get1();

s[i].calculate();

}

cout << "\ne\_no \t e\_name\t des \t bp \t hra \t da \t pf \t np \n";

for (i = 0; i < n; i++)

{ s[i].display(); }

getch();

}

**C++ Multiple Inheritance Example**

#include<iostream.h>

#include<conio.h>

class student {

private:

int rno, m1, m2;

public:

void get() {

cout << "Enter the Roll no :";

cin>>rno;

cout << "Enter the two marks :";

cin >> m1>>m2;

}

};

class sports

{

private:

int sm; // sm = Sports mark

public:

void getsm() {

cout << "\nEnter the sports mark :";

cin>>sm;

}

};

class statement : public student, public sports

{

int tot, avg;

public:

void display() {

tot = (m1 + m2 + sm);

avg = tot / 3;

cout << "\n\n\tRoll No : " << rno << "\n\tTotal : " << tot;

cout << "\n\tAverage : " << avg;

}

};

void main() {

clrscr();

statement obj;

obj.get();

obj.getsm();

obj.display();

getch();

}

**C++ Hierarchical Inheritance**

**Syntax:** Class A{

public void methodA()

{//Do Something }

}

Class B : public A

{

public void methodB()

{//Do Something }

}

Class C : public A

{

public void methodC()

{//Do Something } }

**C++ Multi Level Inheritance**

**Syntax:**  class A

{

... .. ...

};

class B: public A

{

... .. ...

};

class C: public B

{

... ... ...

};

**C++ Polymorphism**

The term "Polymorphism" is the combination of "poly" + "morphs" which means many forms. It is a Greek word.

* **There are two types of polymorphism in C++:**
* **Compile time polymorphism:** It is achieved by function overloading and operator overloading which is also known as static binding or early binding.
* **Runtime polymorphism:** It is achieved by method overriding which is also known as dynamic binding or late binding.

**C++ Overloading (Function and Operator)**

If we create two or more members having same name but different in number or type of parameter, it is known as C++ overloading.

Types of overloading in C++ are:

* **Function overloading**
* **Operators overloading**
* **Function Overloading**

Having two or more function with same name but different in parameters, is known as function overloading in C++.The **advantage** of Function overloading is that it increases the readability of the program because you don't need to use different names for same action.

**Example:**

* int test() { }
* int test(int a) { }
* float test(double a) { }
* int test(int a, double b) { }

**Function Overloading Example**

#include <iostream>

class Cal {

     public:   int add(int a,int b)

{    return (a + b);     }

 int add(int a, int b, int c)

 {     return (a + b + c);    }

};

void  main(void)

{

    Cal C;

    cout<<C.add(10, 20)<<endl;

    cout<<C.add(12, 20, 23);

   getch();

}

* **Operators Overloading**

Operator overloading is used to overload or redefine most of the operators available in C++. It is used to perform operation on user define data type.

The advantage of Operators overloading is to perform different operations on the same operand.

**Following are list of operators, which cannot be overloaded:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **::**  Scope resolution operator | **.\***  Pointer to member operator | **.**  Dot membership operatot | **?:**  Conditional Operator | Sizeof() operator |

**Example:**

#include <iostream.h>

#include<conio.h>

class Test

{

   private:   int num;

   public: void getdata(int x ){num=x;}

   void operator ++()

        {

           num = num+2;

        }

        void Print() {   cout<<"The Count is: "<<num;       }

};

void  main()

{

    Test tt;

clrscr( );

tt.getdata(20);

tt.Print();

    ++tt;  // calling of a function "void operator ++()"

Cout<<”\nafter calling operator function”;

    tt.Print();

    getch( );

}

**Java**

Java was developed by James Gosling and his team , from Sun Microsystems in 1995 as an object-oriented language for general-purpose business applications and for interactive, Web-based Internet applications

**Features of JAVA:**

* **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
* **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
* **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
* **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
* **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
* **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
* **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
* **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
* **Distributed** − Java is designed for the distributed environment of the internet.
* **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

**Tools Need:**

Computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).

also need the following software −

* Linux 7.1 or Windows xp/7/8 operating system
* Java JDK 8
* Microsoft Notepad or any other text editor

|  |  |
| --- | --- |
| **Comparison Between c++ and Java** | |
| **C++** | **Java** |
| C++ is platform-dependent. | Java is platform-independent. |
| C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. |
| C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java. |
| C++ supports operator overloading. | Java doesn't support operator overloading. |
| C++ supports pointers. You can write pointer program in C++. | Java supports pointer internally. But you can't write the pointer program in java. It means java has restricted pointer support in java. |
| C++ uses compiler only. | Java uses compiler and interpreter both. |
| C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |
| C++ supports structures and unions. | Java doesn't support structures and unions. |

How to write the simple program of java. We can write a simple hello java program easily after installing the JDK.

To create a simple java program, you need to create a class that contains main method **in any text editor (notepad,notepad++,wordpad etc)**

**Creating hello java example**

class Simple{

    public static void main(String args[ ]){

     System.out.println("Hello Java");

    }

}

**Program Execution:**

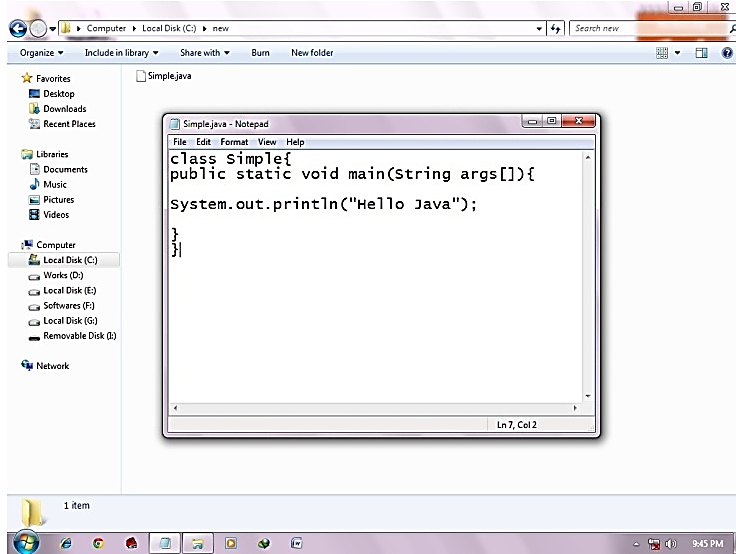
Save the above Program as **Simple.java,** To compile and run this program, you need to open command prompt by **start menu -> All Programs -> Accessories -> command prompt**.

|  |  |
| --- | --- |
| **To compile:** | javac Simple.java |
| **To execute:** | java Simple |

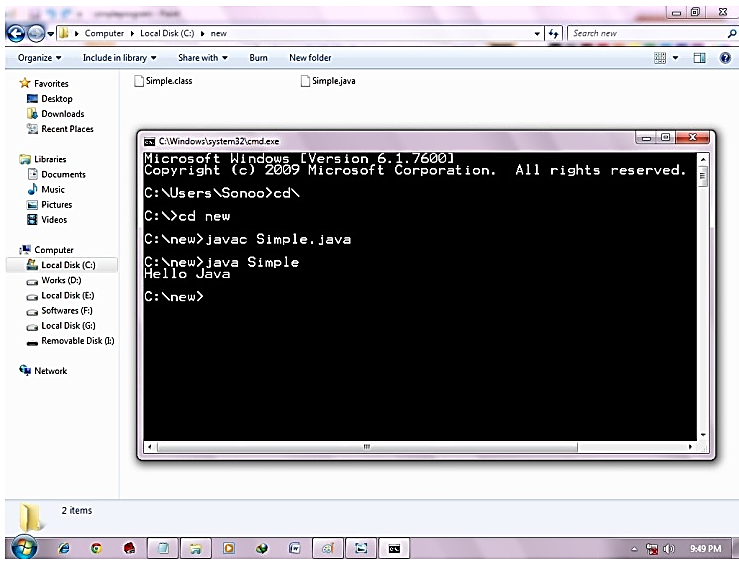
**Parameters used in first java program**

* **class** keyword is used to declare a class in java.
* **public** keyword is an access modifier which represents visibility, it means it is visible to all.
* **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require creating object to invoke the main method. So it saves memory.
* **void** is the return type of the method, it means it doesn't return any value.
* **main** represents the starting point of the program.
* **String[ ] args** is used for command line argument.
* **System.out.println()** is used print statement.

To write the simple program, open notepad by **start menu -> All Programs -> Accessories -> notepad** and write simple program as displayed below:



As displayed in the above diagram, write the simple program of java in notepad and saved it as Simple.java. To compile and run this program, you need to open command prompt by **start menu -> All Programs -> Accessories -> command prompt**.



**Basic Structure of JAVA Program:**

A Java program involves the following sections:

|  |
| --- |
| * Documentation Section * Package Statement * Import Statements * Interface Statement * Class Definition * Main Method Class   + Main Method Definition |

# How to set path in Java:

To set the temporary path of JDK, you need to follow following steps:

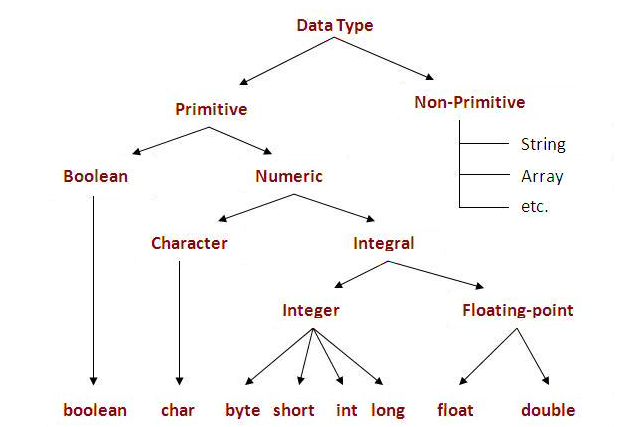
* Open command prompt
* Copy the path of jdk/bin directory
* Write in command prompt: set path=copied\_path

**For Example:**

**set path=C:\Program Files\Java\jdk1.6.0\_23\bin**

# Variables and Data Types in Java

A variable is a container which holds the value while the java program is executed. A variable is assigned with a datatype.

There are two types of data types in java: **primitive and non-primitive.**

# Operators in java:

**Operator** in java is a symbol that is used to perform operations.

There are many types of operators in java which are given below:

* Unary Operator [ ++, -- ]
* Arithmetic Operator, [ +, - , \* , /, %]
* Relational Operator [ >, >=, <, <=, = =,! =]
* Logical Operator [&& , || , !]
* Ternary Operator [ ? :]
* Assignment Operator [ =]

**Java If-else Statement**

The Java *if statement* is used to test the condition. It checks Boolean condition: *true* or *false*. There are various types of if statement in java.

* if statement
* if-else statement
* if-else-if ladder
* nested if statement

## Java if Statement: The Java if statement tests the condition. It executes the if block if condition is true.

**Syntax: if(condition) {   //code to be executed }**

**Example**: public class IfExample

{ public static void main(String[] args)

{ int age=20;

if(age>18){ System.out.print("Eligible to VOTE"); }

} }

## Java if-else Statement:The Java if-else statement also tests the condition. It executes the *if block* if condition is true otherwise *else block* is executed.

**Syntax: if(condition){  //code if condition is true  }**

**else{  //code if condition is false  }**

**Example:** public class IfElseExample {

public static void main(String[] args) {

int number=13;

if(number%2==0){ System.out.println("even number"); }

else{ System.out.println("odd number"); }

} }

## Java if-else-if ladder Statement: The if-else-if ladder statement executes one condition from multiple statements.

**Syntax: if(condition1)**

**{  //code to be executed if condition1 is true**

**}else if(condition2){**

**//code to be executed if condition2 is true**

**}**

**else if(condition3){**

**//code to be executed if condition3 is true**

**}**

**...**

**else{**

**//code to be executed if all the conditions are false**

**}**

**Example:**

public class Test {

public static void main(String args[]) {

int x = 30;

if( x == 10 ) {

System.out.print("Value of X is 10");

}else if( x == 20 ) {

System.out.print("Value of X is 20");

}else if( x == 30 ) {

System.out.print("Value of X is 30");

}else {

System.out.print("This is else statement");

}

}

}

# Java Switch Statement

**Syntax: switch(expression){**

**case value1:**

**//code to be executed;**

**break;**

**case value2:**

**//code to be executed;**

**break;**

**......**

**default:**

**code to be executed if all cases are not matched;**

**}**

**Example:** public class Test {

public static void main(String args[]) {

char grade = 'C';

switch(grade) {

case 'A' :

System.out.println("Excellent!");

break;

case 'B' :

case 'C' :

System.out.println("Well done");

break;

case 'D' :

System.out.println("You passed");

case 'F' :

System.out.println("Better try again");

break;

default :

System.out.println("Invalid grade");

}

System.out.println("Your grade is " + grade);

}}

**Loops in Java**

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in java.

* for loop
* while loop
* do-while loop

**Java For Loop**

**Syntax: for(initialization;condition;incr/decr){**

**//code to be executed**

**}**

**Example:** public class ForExample {

public static void main(String[] args) {

for(int i=1;i<=10;i++){

System.out.println(i);

}

}

}

**Java While Loop**

**Syntax: while(condition){**

**//code to be executed**

**}**

**Example:**

public class WhileExample {

public static void main(String[] args) {

    int i=1;

    while(i<=10){

        System.out.println(i);

    i++;

    }

}

}

**Java Do While Loop**

**Syntax:**

**do{**

**//code to be executed**

**}while(condition);**

**Example:**

public class DoWhileExample {

public static void main(String[] args) {

    int i=1;

    do{

        System.out.println(i);

    i++;

    }while(i<=10);

}

}

# Java Comments

The java comments are statements that are not executed by the compiler and interpreter. The comments can be used to provide information or explanation about the variable, method, class or any statement. It can also be used to hide program code for specific time.

## Types of Java Comments: There are 3 types of comments in java.

1. Single Line Comment //This is single line comment
2. Multi Line Comment

/\* This  is

multi line  comment \*/

1. Documentation Comment

/\*\*

This

is  documentation

comment

\*/

**\*Example Programs:**

1. // Fibonacci series program in java without using recursion

class FibonacciExample{

public static void main(String args[])

{ int n1=0,n2=1,n3,i,count=10;

System.out.print(n1+" "+n2);//printing 0 and 1

for(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed

{ n3=n1+n2;

System.out.print(" "+n3);

n1=n2;

n2=n3;

} }}

2./\* prime number program in java. In this java program, we will take a number variable and check whether the number is prime or not.\*/

public class PrimeExample{

public static void main(String args[]){

int i,m=0,flag=0;

int n=3;//it is the number to be checked

m=n/2;

if(n==0||n==1){

System.out.println(n+" is not prime number");

}else{

for(i=2;i<=m;i++){

if(n%i==0){

System.out.println(n+" is not prime number");

flag=1;

break; } }

if(flag==0) { System.out.println(n+" is prime number"); }

}//end of else

} }

3.// factorial Program using loop in java.

class FactorialExample{

 public static void main(String args[]){

  int i,fact=1;

  int number=5;//It is the number to calculate factorial

  for(i=1;i<=number;i++){

      fact=fact\*i;

  }

  System.out.println("Factorial of "+number+" is: "+fact);

 }

}

## OOPs (Object Oriented Programming System)

**Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

**Object:** Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

**Class: Collection of objects** is called class. It is a logical entity.

**Inheritance: When one object acquires all the properties and behaviors of parent object** i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**Polymorphism:** When **one task is performed by different ways** i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

**Abstraction: Hiding internal details and showing functionality** is known as abstraction. For example: phone call, we don't know the internal processing. In java, we use abstract class and interface to achieve abstraction.

**Encapsulation: Binding (or wrapping) code and data together into a single unit is known as encapsulation**. For example: capsule, it is wrapped with different medicines.

### Class in Java

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

### Syntax to declare a class:

**class <class\_name>{**

**field;**

**method;**

**}**

**Example:** 1)

class Student{

int id =100;//field or data member or instance variable

String name;

public static void main(String args[]){

Student s1=new Student();//creating an object of Student

System.out.println(s1.id);//accessing member through reference variable

System.out.println(s1.name); } }

### Object and Class

### Example: Rectangle

class Rectangle{

int length;

int width;

void insert(int l, int w){

length=l;

width=w;

}

void calculateArea(){System.out.println(length\*width);}

}

class TestRectangle1{

public static void main(String args[]){

Rectangle r1=new Rectangle();

Rectangle r2=new Rectangle();

r1.insert(11,5);

r2.insert(3,15);

r1.calculateArea();

r2.calculateArea();

}

}

# Inheritance in Java

**Inheritance in java** is a mechanism in which one object acquires all the properties and behaviors of parent object. It is an important part of OPPs(Object Oriented programming system).

### Terms used in Inheritence

* **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
* **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
* **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
* **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in previous class.

### Syntax of Deriving a new class:

**class Subclass-name extends Superclass-name**

**{**

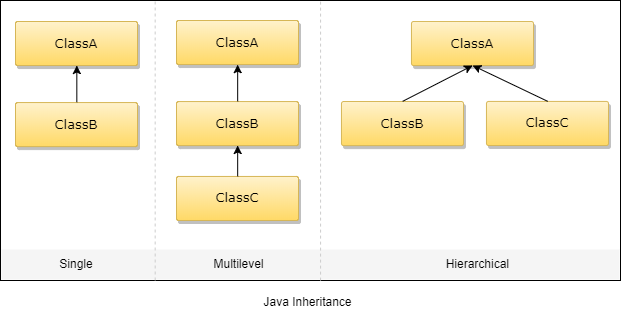
**//methods and fields**

**}**

## Types of inheritance in java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.[In java programming, multiple and hybrid inheritance is supported through interface only.]

***Note: Multiple inheritance is not supported in java through class.***

******

1. **Single Inheritance**

When a single class gets derived from its base class, then this type of inheritance is termed as single inheritance.

**Example:**

class Teacher {

void teach() {

System.out.println("Teaching subjects");

}

}

class Students extends Teacher {

void listen() {

System.out.println("Listening to teacher");

}

}

class CheckForInheritance {

public static void main(String args[]) {

Students s1 = new Students();

s1.teach();

s1.listen();

}

}

1. **Multi-Level Inheritance:** In this type of inheritance, a derived class gets created from another derived class and can have any number of levels.

**Example** : class Shape {

   public void display() {

      System.out.println("Inside display");

   }

}

class Rectangle extends Shape {

   public void area() {

      System.out.println("Inside area");

   }

}

class Cube extends Rectangle {

   public void volume() {

      System.out.println("Inside volume");

   }

}

public class Tester {

   public static void main(String[] arguments) {

      Cube cube = new Cube();

      cube.display();

      cube.area();

      cube.volume();

   }

}

1. **Hierarchical Inheritance:** In this type of inheritance, there are more than one derived classes which get created from one single base class.

**Example:**

class A

{

public void methodA()

{

System.out.println("method of Class A");

}

}

class B extends A

{

public void methodB()

{

System.out.println("method of Class B");

}

}

class C extends A

{

public void methodC()

{

System.out.println("method of Class C");

}

}

class D extends A

{

public void methodD()

{

System.out.println("method of Class D");

}

}

class JavaExample

{

public static void main(String args[])

{

B obj1 = new B();

C obj2 = new C();

D obj3 = new D();

//All classes can access the method of class A

obj1.methodA();

obj2.methodA();

obj3.methodA();

}

}

# Java Package

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

## Simple example of java package

The **package keyword** is used to create a package in java.

**//save as Simple.java**

package mypack;

public class Simple{

 public static void main(String args[]){

    System.out.println("Welcome to package");

   }

}

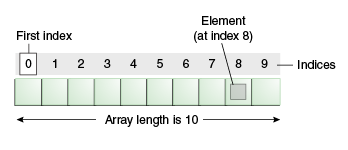
# Access Modifiers in java

**There are 4 types of java access modifiers:**

1. **private:** The private access modifier is accessible only within class.
2. **default:** If you don't use any modifier, it is treated as **default** bydefault. The default modifier is accessible only within package.
3. **protected:** The **protected access modifier** is accessible within package and outside the package but through inheritance only. The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.
4. **public:** The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers.

# Java Array

# Java array is an object the contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array.

Array in java is index based, first element of the array is stored at 0 index.

### Advantage of Java Array

* **Code Optimization:** It makes the code optimized, we can retrieve or sort the data easily.
* **Random access:** We can get any data located at any index position.

### Types of Array in java

There are two types of array.

* Single Dimensional Array
* Multidimensional Array

### Single dimensional array in java

### Syntax to Declare an Array in java

dataType[ ]  arr;  ( or)

dataType  []arr; (or)

dataType  arr[];

### Instantiation of an Array in java

arrayRefVar=new datatype[size];

**Example:**

class Testarray{

public static void main(String args[]){

int a[]=new int[5];//declaration and instantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

//printing array

for(int i=0;i<a.length;i++)//length is the property of array

System.out.println(a[i]);

}}

## Declaration, Instantiation and Initialization of Java Array

class Testarray1{

public static void main(String args[]){

int a[]={33,3,4,5};//declaration, instantiation and initialization

//printing array

for(int i=0;i<a.length;i++)//length is the property of array

System.out.println(a[i]);

}}

### Multidimensional array in java

### Syntax to Declare Multidimensional Array in java

**dataType[ ][ ]  arrayRefVar; (or)**

**dataType  [ ][ ]arrayRefVar; (or)**

**dataType  arrayRefVar[][]; (or)**

### Instantiation of an Multi Dimensional Array in java

**int[][] arr=new int[3][3];//3 row and 3 column**

**Example:**

class Testarray3{

public static void main(String args[]){

//declaring and initializing 2D array

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

//printing 2D array

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

System.out.print(arr[i][j]+" ");

}

System.out.println();

}

}}

# Java String

Generally, string is a sequence of characters. But in java, string is an object that represents a sequence of characters. The java.lang.String class is used to create string object.

**Example:**

1. char[] ch={'j','a','v','a','t','p','o','i','n','t'};
2. String s=new String(ch);

**Java String** class provides a lot of methods to perform operations on string such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

There are two ways to create String object:

* + 1. By string literal🡪 String  s="welcome";
  1. By new keyword🡪 String s=new String("Welcome");

**Example:**

public class StringExample{

public static void main(String args[ ]){

String s1="java";

char ch[ ]={'s','t','r','i','n','g','s'};

String s2=new String(ch);

String s3=new String("example");

System.out.println(s1);

System.out.println(s2);

System.out.println(s3);

}}

# Exception Handling in Java

The **exception handling** in java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.

**Exception Handling** is a mechanism to handle runtime errors such as ClassNotFound, IO etc.

## Java try block

Java try block is used to enclose the code that might throw an exception. It must be used within the method.Java try block must be followed by either catch .

## Syntax of java try-catch

try{

//code that may throw exception

}catch(Exception\_class\_Name ref){ }

**Example:**

public class Testtrycatch2{

public static void main(String args[]){

try{

int data=50/0;

}catch(ArithmeticException e){System.out.println(e);}

System.out.println("rest of the code...");

}

}

**Compile by: javac Testtrycatch2.java**

**Run by: java Testtrycatch2**

java.lang.ArithmeticException: / by zero  
rest of the code...