INDEX

ABSTRACT

Abstract

I have always believed that there is only one way to learn a language: start with “basics of the basics”.

When learning a language, I like to have a best lecturer who teaches every single topic even when I have doubts, and I would like to be explained in detail, but in situations like “lecturer would be busy” or “we cannot contact him/her” I would be in a big trouble. I would like to have a tutor which is similar to a book but more than a book.

Programming Language like “C-language” is the basic and important language among others. Now-a-days, internet is also used for education. Learning through this is easy by which a student can learn languages in his/her free time. In our project, we are providing Language Tutor, a platform for learning programming languages through online.

INTRODUCTION

Introduction

Our intension is to give a helping hand to those who are interested to learn Computer Languages like C-language, HTML, C++, CSS, Java, etc.

We provide C-language in our tutor.

Our project is to provide a website which deals with tutorial through which a user/learner can learn languages. When they get any doubts, they can spontaneously clarify by visiting our website.

Modules

In our project, criteria deals with following modules:

1. Admin Login,
2. User Login, and
3. C-Language.

Contents

As on the part of project, we are creating a website which contains the below contents:

1. History of Language.
2. Basics of Language.
3. Main Topics,
4. Syntax's
5. Example programs
6. Important programs.

SYSTEM REQUIREMENTS

System Requirements

Hardware Requirements:

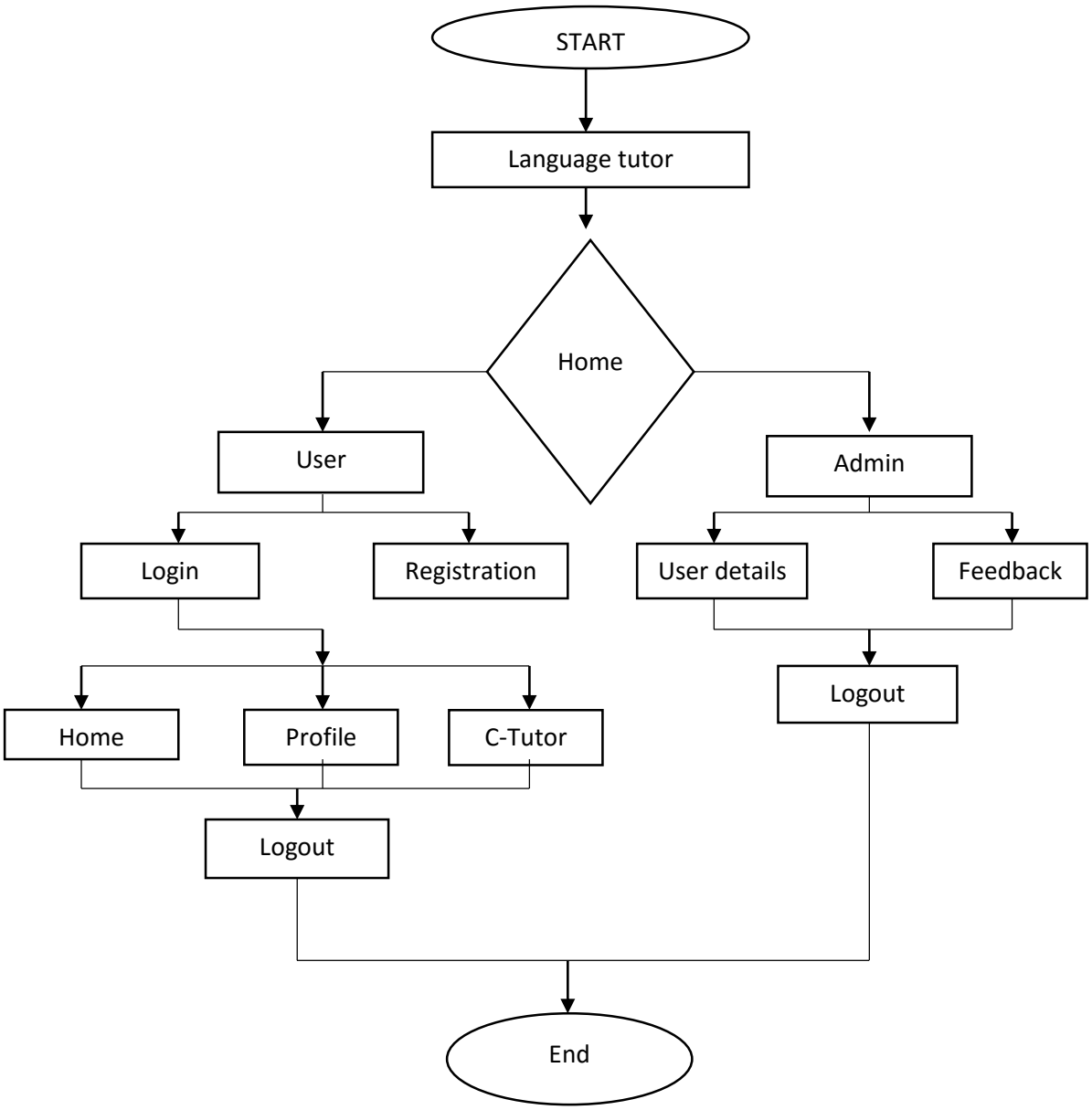
- **Processor** - Pentium-III
- **Speed** - 2.0GHz
- **RAM** - 1 GB

Software Requirements:

- **Operating System** : Windows XP/7/8/8.1/10
- **Application** : Google Chrome, Mozilla Firefox
- **Front End** : HTML, CSS
- **Plug-in languages** : JavaScript
- **Database Connectivity** : MySQL

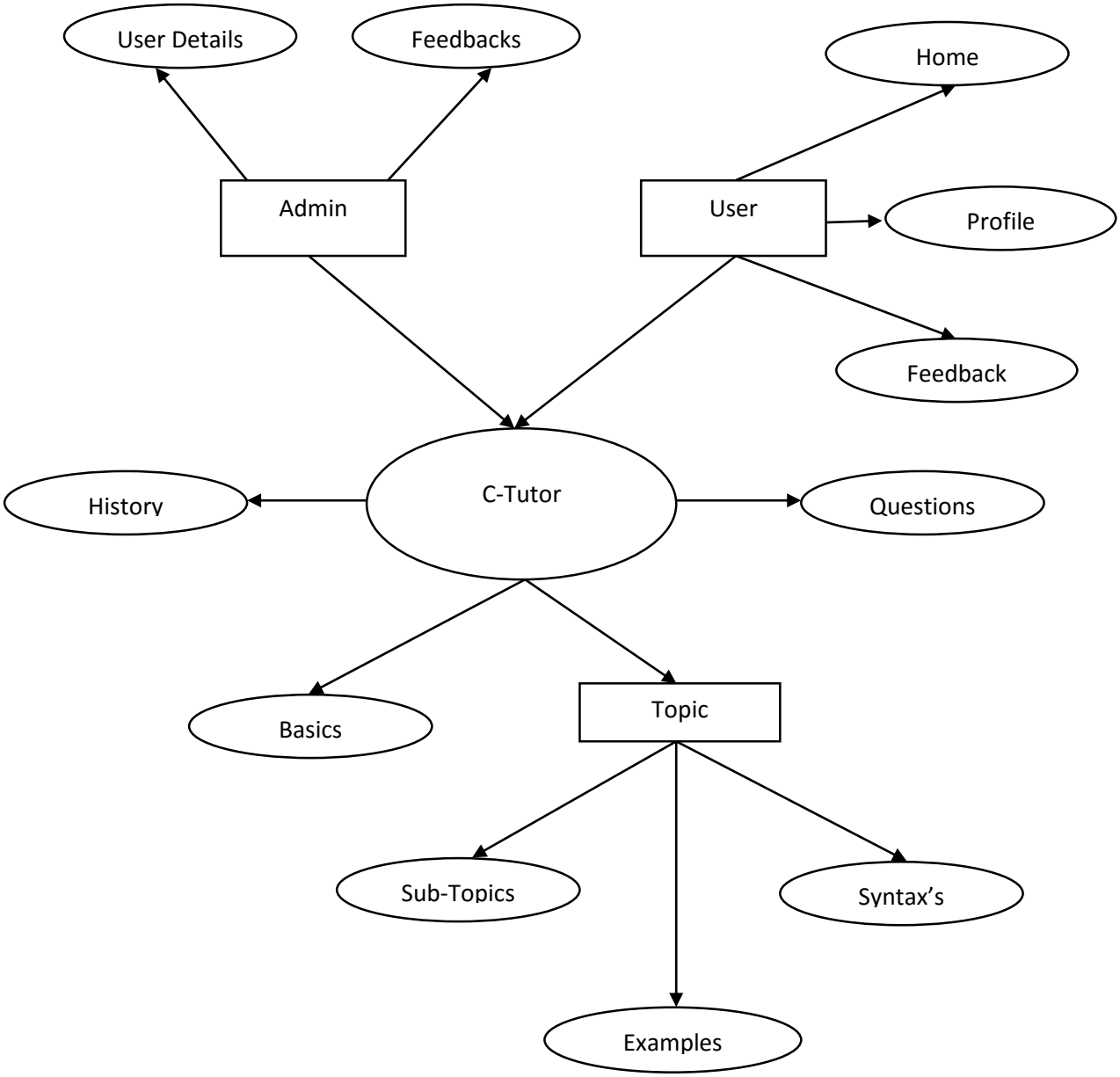
FLOW CHART

Flow Chart


START
Language tutor
Home
User
Admin
Login
Registration
User details
Feedback
Home
Profile
C-Tutor
Logout
Logout
End

E-R DIAGRAM

E-R Diagram


User Details
Feedbacks
Home
Admin
User
Profile
Feedback
C-Tutor
History
Questions
Basics
Topic
Sub-Topics
Syntax's
Examples

Project Code

Home.html

```
<!doctype html>

<html><head>

<meta charset="utf-8">

<title>Home</title>

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

</head><body>

<div id="Container">

 <div id="Header"><img src="Logo/Logo.png" alt="Logo" width="100" height="105" title="Logo"/><img id="titile" src="Logo/title.png" width="800" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="home.html"><span>Home</span></a></li>

  <li class="active has-sub"><a href="Admin_Login.html"><span>Admin Login</span></a>

  </li>

  <li class="active has-sub"><a href="#"><span>User</span></a>

    <ul>

      <li class="has-sub"><a href="user_login.html"><span>login</span></a></li>

      <li class="has-sub"><a href="user_register.html"><span>Register</span></a></li>

    </ul>

  </li>

  <li class="last"><a href="Contact.html"><span>Contact</span></a></li>

</ul>

</div>

 <div id="Content">

            <div style="float:left;margin:1px 1px 1px 1px;height:auto;width:79.7%;">

             <h3 align="center">Abstract</h3>

    <p style="text-align: justify">I have always believed that there is  only one way to learn a language: start with &amp;ldquo;basics of the basics&amp;rdquo;. When learning a language, I like to  have a best lecturer who teaches every single topic even when I have doubts,  and I would like to be explained in detail, but in situations like &amp;ldquo;lecturer  would be busy&amp;rdquo; or &amp;ldquo;we cannot contact him/her&amp;rdquo; I would be in a big trouble.  I would like to have a tutor which is similar  to a book but more than a book.</p>

    <p style="text-align: justify">Programming Language like  &amp;ldquo;C-language&amp;rdquo; is the basic and important language among others. Now-a-days,  internet is also used for education. Learning through this is easy by which a  student can learn languages in his/her free time. In our project, we are  providing Language Tutor, a platform for learning programming languages through  online.</p>

    <h3 align="center">Introduction</h3>

     <p style="text-align: justify">Our intension is to give a helping  hand to those who are interested to learn Computer Languages like C-language,  HTML, C++, CSS, Java, etc.</p>

     <p>We provide C-language in our tutor.</p>
```

```
<p style="text-align: justify">Our project is to provide a website which deals with tutorial through which a user/learner can learn languages. When they get any doubts, they can spontaneously clarify by visiting our website.</p>

<h3 style="text-align:center;">System Requirements</h3>

<h5 style="font-style:italic;">Hardware Requirements:</h5>

<ul>

<li>Processor - Pentium-III</li>

<li>Speed - 2.0 GHz</li>

<li>RAM - 1 GB</li></ul>

<h5 style="font-style:italic;">Software Requirements:</h5>

<ul><li>Operating System - Windows XP/7/8/8.1/10</li>

<li>Application - Google Chrome, Mozilla Firefox</li>

<li>Front End - HTML, CSS</li>

<li>Plug-in Languages - JavaScript</li>

<li>Database Connectivity - MySQL</li></ul>

<h3 align="center">Modules</h3>

<span style="text-align: justify">In our project, criteria deals with following modules:</span>

<ol>

<li>Admin Login,</li>

<li>User Login, and</li>

<li>C-Language.</li>

</ol>

<h3 align="center">Contents</h3>

<span style="text-align: justify">As on the part of project, we are creating a website which contains the below contents:</span>

<ol>

<li>History of Language.</li>

<li>Basics of Language.</li>

<li>Main Topics,</li>

<li>Syntax&rsquo;s</li>

<li>Example programs</li>

<li>Important programs.</li>

</ol>

</div>

<div style="float:right;margin:1px 1px 1px 1px;height:auto;width:20%;"><img src="Logo/sgmgpt-1.png" width="250" align="right" height="374" alt=""/></div>

</div>

</body>

</html>
```

Admin_Login.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Admin Login</title>

<link rel="stylesheet" type="text/css" href="CSS/registration.css">

</head>

<body>

<div id="Container">

 <div id="Header"><img src="Logo/Logo.png" alt="Logo" width="100" height="105" title="Logo"/><img id="titile" src="Logo/title.png" width="800" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="home.html"><span>Home</span></a></li>

  <li class="active has-sub"><a href="Admin_Login.html"><span>Admin Login</span></a></li>

  <li class="active has-sub"><a href="#"><span>User</span></a>

    <ul>

      <li class="has-sub"><a href="user_login.html"><span>login</span></a></li>

      <li class="has-sub"><a href="user_register.html"><span>Register</span></a></li>

    </ul>

  </li>

  <li class="last"><a href="Contact.html"><span>Contact</span></a></li>

</ul>

</div>

 <div id="Content">

  <h5 id="subtitle">Admin Login</h5>

  <table width="100%" border="0" align="center" cellpadding="5" cellspacing="0" id="login_table" style="">

    <tbody>

      <tr>

       <td width="45%" height="24" style="text-align: right"><span style="text-align: left"></span><span style="text-align: center"></span>Admin ID</td>

       <td width="55%" style="text-align: left"><input class="textbar" name="Admin_ID" type="text" id="Admin_ID" autofocus size="20" maxlength="20"></td>

      </tr>

      <tr>
```

```
        <td style="text-align: right">Password</td>

        <td style="text-align: left"><p>

          <input class="textbar" name="Admin_Password" type="password" id="Admin_Password" size="20" maxlength="25">

        </p></td>

      </tr>

      <tr>

        <td style="text-align: right"></td>

        <td id="Forgot_Password" style="text-align: left"><input class="buttons" name="Log In" type="button" id="Log In" title="Log In"
value="Log In" alt="log in">

          <input class="buttons" name="Reset_Password" type="button" id="Reset_Password" tabindex="ad" title="Forgot Password? Click
Here" value="Reset Password"></td>

      </tr>

    </tbody>

  </table>

 </div>

 <div id="Footer">Content for id "Footer"</div>

</div>

</body>

</html>
```

User_Register.html

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>User Register</title>
<link rel="stylesheet" type="text/css" href="CSS/registration.css">
<script>
		function enablespecify()
		{
			document.getElementById("Specify2").disabled=false;
			document.getElementById("Specify2").required=true;
		}
		function disablespecify()
		{
			document.getElementById("Specify2").disabled=true;
			document.getElementById("Specify2").value="";
			document.getElementById("Specify2").required=false;
		}

		function verifypassword()
		{
			var a=document.getElementById("password1").value;
			var b=document.getElementById("password2").value;
			if(a.length <8)
			{
				alert("password should be minimum 8 characters.");
				form.password1.focus();
				return false;
			}
			if(a != b)
			{
				alert("password doesn't match");
```

```
                    return false;
                }
                return false;
            }
</script>
</head>
<body>
<div id="Container">
 <div id="Header"><img src="Logo/Logo.png" alt="Logo" width="100" height="105" title="Logo"/><img id="titile" src="Logo/title.png" width="800" height="110" alt="Title"/></div>
 <div id="cssmenu">
<ul>
  <li><a href="home.html"><span>Home</span></a></li>
  <li class="active has-sub"><a href="Admin_Login.html"><span>Admin Login</span></a></li>
  <li class="active has-sub"><a href="#"><span>User</span></a>
    <ul>
      <li class="has-sub"><a href="user_login.html"><span>login</span></a></li>
      <li class="has-sub"><a href="user_register.html"><span>Register</span></a></li>
    </ul>
  </li>
  <li class="last"><a href="Contact.html"><span>Contact</span></a></li>
</ul>
</div>
 <div id="Content">
  <h2 id="subtitle">User Register</h2>
  <form tag="Create_logon">
            <div align="center">
              <table width="100%" height="445" border="0" cellpadding="5">
                <tbody>
                  <tr>
                    <td width="498" style="text-align: right"><label for="username">User Name:</label></td>
                    <td width="547" style="text-align: left"><input class="textbar" name="username" type="text" autofocus required id="username"></td>
        </tr>
                  <tr>
                    <td height="42" style="text-align: right"><label for="email3">Email:</label></td>
                    <td style="text-align: left"><input  class="textbar" name="email" type="email" required id="email3"></td>
```

```
</tr>
        <tr>
          <td style="text-align: right"><label for="userID3">User Id:</label></td>
          <td style="text-align: left"><input class="textbar" name="userID" type="text" required id="userID3"></td>
</tr>
        <tr>
          <td style="text-align: right"><label for="password1">Password:</label></td>
          <td style="text-align: left"><input name="password1" type="password" autofocus required  class="textbar"
id="password1" >
           <input name="showpassword1" type="checkbox" id="showpassword1" title="showpassword1" value="showpassword"
onChange="document.getElementById('password1').type = this.checked ? 'text' : 'password';">
          Show Password</td>
</tr>
        <tr>
          <td style="text-align: right"><label for="password2">Retype  Password:</label></td>
          <td style="text-align: left"><input name="password2" type="password" autofocus required class="textbar"
id="password2">
    <input name="showpassword2" type="checkbox" id="showpassword2" title="showpassword2" value="showpassword"
onChange="document.getElementById('password2').type = this.checked ? 'text':'password';">Show Password
    </td>
</tr>
        <tr>
          <td style="text-align: right"><label for="dob2">Date of Birth</label></td>
          <td style="text-align: left"><input class="textbar" name="dob" type="date" required id="dob2"></td>
</tr>
        <tr>
          <td style="text-align: right"><label>Gender:
    </label></td>
          <td style="text-align: left"><label>
            <input name="Gender" type="radio" id="Gender_0" value="Male">
            Male</label>
      <label>
        <input name="Gender" type="radio" id="Gender_1" value="Female">
    Female</label></td>
</tr>
        <tr>
          <td style="text-align: right"><label for="mobile1">Mobile:</label></td>
```

```
                    <td style="text-align: left"><input class="textbar" name="mobile" type="tel" required id="mobile1"
maxlength="10"></td>

          </tr>

                    <tr>

                     <td style="text-align: right"><p>

              <label>Who are you: </label>

           </p></td>

                     <td style="text-align: left"><label>

                      <input name="WhatAreYou" type="radio" id="WhatAreYou_0" value="Student" onFocus="disablespecify()">

                      Student</label>

             <label>

              <input name="WhatAreYou" type="radio" id="WhatAreYou_1" onFocus="disablespecify()" value="Employee">

              Employee</label>

             <label>

              <input name="WhatAreYou" type="radio" id="WhatAreYou_2" onFocus="disablespecify()" value="Bussiness">

              Bussiness</label>

             <label>

              <input name="WhatAreYou" type="radio" id="WhatAreYou_3" onFocus="enablespecify()" value="Others">

           Others</label></td>

          </tr>

                    <tr>

                     <td style="text-align: right"><label for="Specify2">Specify(<span style="font-weight: normal">If
Others</span>):</label></td>

                     <td style="text-align: left"><textarea  class="textbar" name="Specify" rows="1" maxlength="20" disabled="disabled"
wrap="soft" id="Specify2" title="Specify"></textarea></td>

          </tr>

                    <tr>

                     <td style="text-align: right">

                      <input class="buttons" name="Reset" type="reset" id="Reset" title="Reset" value="Reset">

                      <span style="text-align: right"></span></td>

                     <td style="text-align: left">

                      <span style="text-align: left"></span>

                      <input class="buttons" name="SignUp" type="submit" id="SignUp" formmethod="post" title="SignUp" value="Sign Up"
onClick="verifypassword()">

          </td>         </tr>        </tbody>      </table>     </div>   </form>  </div>

    <div id="Footer">Content for  id "Footer"

  </div></div></body></html>
```

User_Login.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>User Login</title>

<link rel="stylesheet" type="text/css" href="CSS/registration.css">

</head>

<body>

<div id="Container">

 <div id="Header"><img src="Logo/Logo.png" alt="Logo" width="100" height="105" title="Logo"/><img id="titile" src="Logo/title.png" width="800" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="home.html"><span>Home</span></a></li>

  <li class="active has-sub"><a href="Admin_Login.html"><span>Admin Login</span></a></li>

  <li class="active has-sub"><a href="#"><span>User</span></a>

    <ul>

      <li class="has-sub"><a href="user_login.html"><span>login</span></a></li>

      <li class="has-sub"><a href="user_register.html"><span>Register</span></a></li>

    </ul>

  </li>

  <li class="last"><a href="Contact.html"><span>Contact</span></a></li>

</ul>

</div>

 <div id="Content">

   <h5 id="subtitle">User Login</h5>

   <table width="100%" border="0" align="center" cellpadding="5" cellspacing="0">

     <tbody>

       <tr>

        <td width="45%" height="24" style="text-align: right"><span style="text-align: left"></span><span style="text-align: left"></span>User ID</td>

        <td width="55%" style="text-align: left"><input class="textbar" name="User_ID" type="text" id="User_ID" autofocus size="20" maxlength="20"></td>

       </tr>

       <tr>
```

```
        <td style="text-align: right"> Password</td>

        <td style="text-align: left"><p>

           <input class="textbar" name="User_Password" type="password" id="User_Password" size="20" maxlength="25">

        </p></td>

       </tr>

       <tr>

        <td style="text-align: right"></td>

        <td id="Forgot_Password" style="text-align: left">

        <a href="User_Login/welcome.html"><input class="buttons" name="Log In" type="button" id="Log In" title="Log In" value="Log In"
alt="log in">

          <input class="buttons" name="Reset_Password" type="button" id="Reset_Password" tabindex="ad" title="Forgot Password? Click
Here" value="Reset Password"></td></a>

       </tr>

     </tbody>

   </table>

 </div>

 <div id="Footer">Content for  id "Footer"</div>

</div>

</body>

</html>
```

Contact.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Contact</title>

<link rel="stylesheet" type="text/css" href="CSS/contact.css">

</head>

<body>

<div id="Container">

 <div id="Header"><img src="Logo/Logo.png" alt="Logo" width="100" height="105" title="Logo"/><img id="titile" src="Logo/title.png" width="800" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="home.html"><span>Home</span></a></li>

  <li class="active has-sub"><a href="Admin_Login.html"><span>Admin Login</span></a>


  </li>

  <li class="active has-sub"><a href="#"><span>User</span></a>

    <ul>

      <li class="has-sub"><a href="user_login.html"><span>login</span></a></li>

      <li class="has-sub"><a href="user_register.html"><span>Register</span></a></li>

    </ul>

  </li>

  <li class="last"><a href="Contact.html"><span>Contact</span></a></li>

</ul>

</div>

 <div id="Content">

            <div id="subtitle">Contact Us</div>

  <table width="900px" border="1" cellspacing="0" cellpadding="10" align="center">

    <tbody>

      <tr>

       <th width="6%" scope="col">Sl. No.</th>

       <th width="24%" scope="col">Name</th>

       <th width="18%" scope="col">Pin Number</th>
```

```
 <th width="15%" scope="col">Mobile Number</th>
 <th width="37%" scope="col">Email</th>
</tr>
<tr>
 <td style="font-weight: bold; text-align: center;">1</td>
 <td>Gnaneshwar Gajwel</td>
 <td>13047-CM-052</td>
 <td>7207 673 483</td>
 <td style="text-align: center">nanisonu22@gmail.com</td>
</tr>
<tr>
 <td style="font-weight: bold; text-align: center;">2</td>
 <td>Arshaque Mohammed</td>
 <td>13047-CM-027</td>
 <td>7416 580 623</td>
 <td style="text-align: center">mohd.arshaque@gmail.com</td>
</tr>
<tr>
 <td style="font-weight: bold; text-align: center;">3</td>
 <td>Dosakayala Nikhil Duth</td>
 <td>13047-CM-038</td>
 <td>8977 405 878</td>
 <td style="text-align: center">nikhilduth2@gmail.com</td>
</tr>
<tr>
 <td style="font-weight: bold; text-align: center;">4</td>
 <td>Sudhir Kumar</td>
 <td>13047-CM-040</td>
 <td>7093 595 363</td>
 <td style="text-align: center">sudhir.m.s.d.0001@gmail.com </td>
</tr>
<tr>
 <td style="font-weight: bold; text-align: center;">5</td>
 <td>Aziz Mohammed</td>
 <td>13047-CM-035</td>
 <td>8099 842 426</td>
```

```
      <td style="text-align: center">-</td>
     </tr>
     <tr>
      <td style="font-weight: bold; text-align: center;">6</td>
      <td>Srikanth</td>
      <td>13047-CM-049</td>
      <td>9010 066 404</td>
      <td> </td>
     </tr>
     <tr>
      <td style="font-weight: bold; text-align: center;">7</td>
      <td>Dayyala Shiva Kumar</td>
      <td>13047-CM-041</td>
      <td>9666 453 356</td>
      <td> </td>
     </tr>
    </tbody>
   </table>
  </div>
  <div id="Footer">Content for  id "Footer" Goes Here</div>
</div>
</body>
</html>
```

Welcome.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Welcome</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content">

   <ul>

    <li>The C programming language is a structure oriented programming language, developed at Bell Laboratories in 1972 by Dennis
Ritchie</li>
```

C programming language features were derived from an earlier language called “B” (Basic Combined Programming Language – BCPL)

C language was invented for implementing UNIX operating system

In 1978, Dennis Ritchie and Brian Kernighan published the first edition “The C Programming Language” and commonly known as K&R C

In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI standard, or “ANSI C”, was completed late 1988.

<h3>C programming language standards:</h3>

C89/C90 standard – First standardized specification for C language was developed by the American National Standards Institute in 1989. C89 and C90 standards refer to the same programming language.

C99 standard – Next revision was published in 1999 that introduced new features like advanced data types and other changes.

<h3>C11 and Embedded C language:</h3>

C11 standard adds new features to C programming language and library like type generic macros, anonymous structures, improved Unicode support, atomic operations, multi-threading and bounds-checked functions. It also makes some portions of the existing C99 library optional and improves compatibility with C++.

Embedded C includes features not available in C like fixed-point arithmetic, named address spaces, and basic I/O hardware addressing.

Operating systems, C compiler and all UNIX application programs are written in C language

It is also called as procedure oriented programming language. The C language is reliable, simple and easy to use. C has been coded in assembly language.

<h3>Features of C programming language:</h3>

Reliability

Portability

Flexibility

Interactivity

Modularity

Efficiency and Effectiveness

<h3>Uses of C programming language:</h3>

The C programming language is used for developing system applications that forms a major portion of operating systems such as Windows, UNIX and Linux. Below are some examples of C being used.

Database systems

Graphics packages

```
 <li>Word processors</li>
 <li>Spreadsheets</li>
 <li>Operating system development</li>
 <li>Compilers and Assemblers</li>
 <li>Network drivers</li>
 <li>Interpreters</li>
</ul>
<h3><strong>Which level is C language belonging to?</strong></h3>
<table width="900" border="1">
 <tbody>
  <tr style="font-weight:bold">
   <td width="34">S.no</td>
   <td width="299"><div>High Level</div></td>
   <td width="297"><div>Middle Level</div></td>
   <td width="242"><div>Low Level</div></td>
  </tr>
  <tr>
   <td>1</td>
   <td>High level languages provide almost everything that the programmer might need to do as already built into the language</td>
   <td>Middle level languages don&rsquo;t provide all the built-in functions found in high level languages, but provides all building blocks that we need to produce the result we want</td>
   <td>Low level languages provides nothing other than access to the machines basic instruction set</td>
  </tr>
  <tr>
   <td>2</td>
   <td>Examples:<br>
    Java, Python</td>
   <td>C, C++</td>
   <td>Assembler</td>
  </tr>
 </tbody>
</table>
<h3><strong>The C language is a structured language</strong></h3>
<table width="900" border="2" style="">
 <tbody>
  <tr style="font-weight:bold">
   <td width="34">S.no</td>
```

```
 <td width="306"><div>Structure oriented</div></td>
 <td width="298"><div>Object oriented</div></td>
 <td width="242"><div>Non structure</div></td>
</tr>
<tr>
 <td>1</td>
 <td>In this type of language, large programs are divided into small programs called functions</td>
 <td>In this type of language, programs are divided into objects</td>
 <td>There is no specific structure for programming this language</td>
</tr>
<tr>
 <td>2</td>
 <td>Prime focus is on functions and procedures that operate on the data</td>
 <td>Prime focus is in the data that is being operated and not on the functions or procedures</td>
 <td>N/A</td>
</tr>
<tr>
 <td>3</td>
 <td>Data moves freely around the systems from one function to another</td>
 <td>Data is hidden and cannot be accessed by external functions</td>
 <td>N/A</td>
</tr>
<tr>
 <td>4</td>
 <td>Program structure follows &amp;ldquo;Top Down Approach&amp;rdquo;</td>
 <td>Program structure follows &amp;ldquo;Bottom UP Approach&amp;rdquo;</td>
 <td>N/A</td>
</tr>
<tr>
 <td>5</td>
 <td><div>Examples:</div>
  <div>C, Pascal, ALGOL and Modula-2</div></td>
 <td><div></div>
  <div>C++, JAVA and C# (C sharp)</div>
  <div></div></td>
 <td> BASIC, COBOL, FORTRAN</td>
```

```
      </tr>
    </tbody>
  </table>
  <h3><strong>Key points to remember in C language:</strong></h3>
  <ol>
    <li>The C language is structured, middle level programming language developed by Dennis Ritchie</li>
    <li>Operating system programs such as Windows, Unix, Linux are written in C language</li>
    <li>C89/C90 and C99 are two standardized editions of C language</li>
    <li>C has been written in assembly language</li>
  </ol>
  <h3><strong>C language tutorial reference E-books &amp;amp; research papers:</strong></h3>
  <ul>
    <li>[ANSI 89] American National Standards Institute., American National Standard for Information  Programming Language C, X3 159-1989</li>
    <li>[Kernighan 78] B. W. Kernighan and D. M. Ritchie, The C Programming Language, Prentice-Hall: Englewood Cliffs, NJ, 1978. Second edition, 1988.</li>
    <li>[Thinking 90] C* Programming Guide, Thinking Machines Corp.: Cambridge Mass., 1990.</li>
  </ul>
 </div>
 <div id="Footer">Content for  id "Footer" Goes Here</div>
</div>
</body>
</html>
```

Basics.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Basics</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content">

   <p>Lots of people get into programming because they love the challenge, are excited by computers and want to build a career creating
web sites, mobile apps or desktop programs. But even if you don&rsquo;t want to become a programmer for a living, it&rsquo;s still worth
your time to learn how to program. I mean this in all seriousness: if computers are at all a part of your life, then<strong> learning to
program is going to improve your life.</strong></p>
```

C is a general-purpose high level language that was originally developed by Dennis Ritchie for the Unix operating system. It was first implemented on the Digital Eqquipment Corporation PDP-11 computer in 1972.

The Unix operating system and virtually all Unix applications are written in the C language. C has now become a widely used professional language for various reasons.

Easy to learn

Structured language

It produces efficient programs.

It can handle low-level activities.

It can be compiled on a variety of computers.

<h2>Facts about C</h2>

C was invented to write an operating system called UNIX.

C is a successor of B language which was introduced around 1970

The language was formalized in 1988 by the American National Standard Institue (ANSI).

By 1973 UNIX OS almost totally written in C.

Today C is the most widely used System Programming Language.

Most of the state of the art software have been implemented using C

<h2>Why to use C ?</h2>

C was initially used for system development work, in particular the programs that make-up the operating system. C was adoped as a system development language because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

Operating Systems

Language Compilers

Assemblers

Text Editors

Print Spoolers

Network Drivers

Modern Programs

Data Bases

Language Interpreters

Utilities

<h2>C Program File</h2>

All the C programs are writen into text files with extension ".c" for example hello.c. You can use "vi" editor, "TurboC" editor, etc., to write your C program into a file.

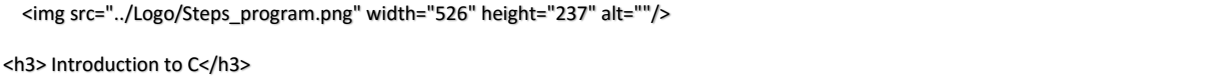
This tutorial assumes that you know how to edit a text file and how to write programming insturctions inside a program file.

<h2>C Compilers</h2>

When you write any program in C language then to run that program you need to compile that program using a C Compiler which converts your program into a language understandable by a computer. This is called machine language (ie. binary format). So before proceeding, make sure you have C Compiler available at your computer. It comes alongwith all flavors of Unix and Linux.

If you are working over Unix or Linux then you can type gcc -v or cc -v and check the result. You can ask your system administrator or you can take help from anyone to identify an available C Compiler at your computer.

<h4>How a C-program works:</h4>



<h3> Introduction to C</h3>

Every full C program begins inside a function called "main". A function is simply a collection of commands that do "something". The main function is always called when the program first executes. From main, we can call other functions, whether they be written by us or by others or use built-in language features. To access the standard functions that comes with your compiler, you need to include a header with the #include directive. What this does is effectively take everything in the header and paste it into your program. Let's look at a working program:

#include <stdio.h>

int main()

{

printf("Hello World!");/* It gives output "Hello World!" to the user*/

getch();

return 0;

}

<table width="800" border="1" cellspacing="0" cellpadding="10px" align="center">

<tbody>

<tr>

<th width="52" scope="col">Sl.No.</th>

<th width="210" scope="col">Command</th>

<th width="470" scope="col">Explanation</th>

</tr>

<tr>

<th scope="row">1</th>

<td>#include <stdio.h></td>

<td> This is a preprocessor command that includes standard input output header file(stdio.h) from the C library before compiling a C program</td>

</tr>

<tr>

```
    <th scope="row">2</th>
    <td>int main()</td>
    <td>This is the main function from where execution of any C program begins.</td>
   </tr>
   <tr>
    <th scope="row">3</th>
    <td>{</td>
    <td>This indicates the beginning of the main function.</td>
   </tr>
   <tr>
    <th scope="row">4</th>
    <td>/*_some_comments_*/</td>
    <td>whatever is given inside the command &amp;ldquo;/*  */&amp;rdquo; in any C program, won&amp;rsquo;t be considered for compilation and execution.</td>
   </tr>
   <tr>
    <th scope="row">5</th>
    <td>printf(&amp;ldquo;Hello_World! &amp;ldquo;);</td>
    <td>printf command prints the output onto the screen.</td>
   </tr>
   <tr>
    <th scope="row">6</th>
    <td>getch();</td>
    <td>This command waits for any character input from keyboard.</td>
   </tr>
   <tr>
    <th scope="row">7</th>
    <td>return 0;</td>
    <td>This command terminates C program (main function) and returns 0.</td>
   </tr>
   <tr>
    <th scope="row">8</th>
    <td>}</td>
    <td>This indicates the end of the main function.</td>
   </tr>
  </tbody>
</table>
```

Finally, at the end of the program, we return a value from main to the operating system by using the return statement. This return value is important as it can be used to tell the operating system whether our program succeeded or not. A return value of 0 means success.

The final brace closes off the function. You should try compiling this program and running it. You can cut and paste the code into a file, save it as a .c file, and then compile it. If you are using a command-line compiler, such as Borland C++ 5.5, you should read the compiler instructions for information on how to compile. Otherwise compiling and running should be as simple as clicking a button with your mouse (perhaps the "build" or "run" button).

You might start playing around with the printf function and get used to writing simple C programs.

Content for id "Footer" Goes Here

</body>

</html>

Topics.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Topics</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content">

 <ul>

 <li><a href="topics/PrintfAndScanfFunctions.html">Printf And Scanf Functions</a></li>
```

```
  <li><a href="topics/DataTypes.html">Data Types</a></li>
  <li><a href="topics/Constants.html">Constants</a></li>
            <li><a href="topics/Variables.html">Variables</a></li>
  <li><a href="topics/OperatorsAndExpressions.html">Operators And Expressions</a></li>
            <li><a href="topics/DecisionControlStatements.html">Decision Control Statements</a></li>
  <li><a href="topics/LoopControlStatements.html">Loop Control Statements</a></li>
            <li><a href="topics/CaseControlStatements.html">Case Control Statements</a></li>
  <li><a href="topics/Arrays.html">Arrays</a></li>
  <li><a href="topics/String.html">String</a></li>
  <li><a href="topics/Pointers.html">Pointers</a></li>
  <li><a href="topics/Functions.html">Functions</a></li>
            <li><a href="topics/Structures.html">Structures</a></li>
 </ul>
 </div>
 <div id="Footer">Content for  id "Footer" Goes Here</div>
</div>
</body>
</html>
```

Contacts.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Contacts</title>

<link rel="stylesheet" type="text/css" href="../CSS/contact.css">

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content">

            <div id="subtitle">Contact Us</div>

  <table width="900px" border="1" cellspacing="0" cellpadding="10" align="center">
```

```
<tbody>
 <tr>
  <th width="6%" scope="col">Sl. No.</th>
  <th width="24%" scope="col">Name</th>
  <th width="18%" scope="col">Pin Number</th>
  <th width="15%" scope="col">Mobile Number</th>
  <th width="37%" scope="col">Email</th>
 </tr>
 <tr>
  <td style="font-weight: bold; text-align: center;">1</td>
  <td>Gnaneshwar Gajwel</td>
  <td>13047-CM-052</td>
  <td>7207 673 483</td>
  <td style="text-align: center">nanisonu22@gmail.com</td>
 </tr>
 <tr>
  <td style="font-weight: bold; text-align: center;">2</td>
  <td>Arshaque Mohammed</td>
  <td>13047-CM-027</td>
  <td>7416 580 623</td>
  <td style="text-align: center">mohd.arshaque@gmail.com</td>
 </tr>
 <tr>
  <td style="font-weight: bold; text-align: center;">3</td>
  <td>Dosakayala Nikhil Duth</td>
  <td>13047-CM-038</td>
  <td>8977 405 878</td>
  <td style="text-align: center">nikhilduth2@gmail.com</td>
 </tr>
 <tr>
  <td style="font-weight: bold; text-align: center;">4</td>
  <td>Sudhir Kumar</td>
  <td>13047-CM-040</td>
  <td>7093 595 363</td>
  <td style="text-align: center">sudhir.m.s.d.0001@gmail.com </td>
 </tr>
```

```
      <tr>
       <td style="font-weight: bold; text-align: center;">5</td>
       <td>Aziz Mohammed</td>
       <td>13047-CM-035</td>
       <td>8099 842 426</td>
       <td style="text-align: center">-</td>
      </tr>
      <tr>
       <td style="font-weight: bold; text-align: center;">6</td>
       <td>Srikanth</td>
       <td>13047-CM-049</td>
       <td>9010 066 404</td>
       <td> </td>
      </tr>
      <tr>
       <td style="font-weight: bold; text-align: center;">7</td>
       <td>Dayyala Shiva Kumar</td>
       <td>13047-CM-041</td>
       <td>9666 453 356</td>
       <td> </td>
      </tr>
    </tbody>
  </table>
 </div>
 </div>
 <div id="Footer">Content for  id "Footer" Goes Here</div>
</div>
</body>
</html>
```

PrintfAndScanfFunctions.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Printf And Scanf Functions</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content">

 <h1 style="text-align:center;">Printf And Scanf Functions</h1>

            <ul>
```

printf() and scanf() functions are inbuilt library functions in C which are available in C library by default. These functions are declared and related macros are defined in “stdio.h” which is a header file.

We have to include “stdio.h” file as shown in below C program to make use of these printf() and scanf() library functions.

<h4> C printf() function:

</h4>

printf() function is used to print the “character, string, float, integer, octal and hexadecimal values” onto the output screen.

We use printf() function with %d format specifier to display the value of an integer variable.

Similarly %c is used to display character, %f for float variable, %s for string variable, %lf for double and %x for hexadecimal variable.

To generate a newline,we use “\n” in C printf() statement.

Note:

C language is case sensitive. For example, printf() and scanf() are different from Printf() and Scanf(). All characters in printf() and scanf() functions must be in lower case.

<h4>Example program for C printf() function:</h4>

#include<stdio.h>

```
int main()
{
        char ch = 'A';
        char str[20] = "sgmlanguagetutor.edu";
        float flt = 10.234;
        int no = 150;
        double dbl = 20.123456;
        printf("Character is %c \n", ch);
        printf("String is %s \n" , str);
        printf("Float value is %f \n", flt);
        printf("Integer value is %d\n" , no);
        printf("Double value is %lf \n", dbl);
        printf("Octal value is %o \n", no);
        printf("Hexadecimal value is %x \n", no);
        return 0;
}
```

Output:

Character is A

String is sgmlanguagetutor.edu

Float value is 10.234000

Integer value is 150

Double value is 20.123456

Octal value is 226

Hexadecimal value is 96

You can see the output with the same data which are placed within the double quotes of printf statement in the program except

%d got replaced by value of an integer variable (no),

%c got replaced by value of a character variable (ch),

%f got replaced by value of a float variable (flt),

%lf got replaced by value of a double variable (dbl),

%s got replaced by value of a string variable (str),

%o got replaced by a octal value corresponding to integer variable (no),

%x got replaced by a hexadecimal value corresponding to integer variable

\n got replaced by a newline.

<h4>C scanf() function:</h4>

scanf() function is used to read character, string, numeric data from keyboard

Consider below example program where user enters a character. This value is assigned to the variable “ch” and then displayed.

Then, user enters a string and this value is assigned to the variable ”str” and then displayed.

<h4>Example program for printf() and scanf() functions in C:</h4>

#include <stdio.h>

int main()

{

char ch;

char str[100];

printf(“Enter any character \n”);

scanf(“%c”, &ch);

printf(“Entered character is %c \n”, ch);

```
            printf(&amp;ldquo;Enter any string ( upto 100 character ) \n&amp;rdquo;);
            scanf(&amp;ldquo;%s&amp;rdquo;, &amp;amp;str);
            printf(&amp;ldquo;Entered string is %s \n&amp;rdquo;, str);
}
```

```
  </div>
  <strong>Output:</strong>
  <div class="output">Enter any character
a
Entered character is a
Enter any string ( upto 100 character )
hai
Entered string is hai</div>
  <ul>
   <li>The format specifier %d is used in scanf() statement. So that, the value entered is received as an integer.
    <ul>
     <li>%c for character variable  (ch),</li>
     <li>%f for float variable  (flt),</li>
     <li>%lf for double variable  (dbl),</li>
     <li>%s for string variable  (str),</li>
    </ul>
   </li>
   <li>Ampersand(&amp;amp;) is used before variable name &amp;ldquo;ch&amp;rdquo; in scanf() statement as &amp;amp;ch.
    <ul>
     <li>It is just like in a pointer which is used to point to the variable.</li>
    </ul>
   </li>
  </ul>
 </div>
 <div id="Footer">
  <ul>
   <li><a href="#"><<</a></li>
   <li><a href="PrintfAndScanfFunctions.html">1</a></li>
   <li><a href="DataTypes.html">2</a></li>
   <li><a href="Constants.html">3</a></li>
   <li><a href="Variables.html">4</a></li>
   <li><a href="OperatorsAndExpressions.html">5</a></li>
```

```
      <li><a href="DecisionControlStatements.html">6</a></li>
      <li><a href="LoopControlStatements.html">7</a></li>
      <li><a href="CaseControlStatements.html">8</a></li>
      <li><a href="Arrays.html">9</a></li>
      <li><a href="String.html">10</a></li>
      <li><a href="Pointers.html">11</a></li>
      <li><a href="Functions.html">12</a></li>
      <li><a href="Structures.html">13</a></li>
      <li><a href="DataTypes.html">>></a></li>
    </ul>
</div>
</div>
</body>
</html>
```

DataTypes.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Data Types</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content">

 <h1 style="text-align:center;">Data Type</h1>

   <ul>
```

```
 <li>C data types are defined as the data storage format that a variable can store a data to perform a specific operation.</li>
 <li>Data types are used to define a variable before to use in a program.</li>
 <li>Size of variable, constant and array are determined by data types.</li>
</ul>
<h4><strong>C – data types:<br>
</strong></h4>
<p>There are four data types in C language. They are,</p>
<table width="500" border="1">
 <tbody>
  <tr>
   <td width="34">S.no</td>
   <td width="184"><div>Types</div></td>
   <td width="266"><div>Data Types</div></td>
  </tr>
  <tr>
   <td>1</td>
   <td>Basic data types</td>
   <td>int, char, float, double</td>
  </tr>
  <tr>
   <td>2</td>
   <td>Enumeration data type</td>
   <td>enum</td>
  </tr>
  <tr>
   <td>3</td>
   <td>Derived data type</td>
   <td>pointer, array, structure, union</td>
  </tr>
  <tr>
   <td>4</td>
   <td>Void data type</td>
   <td>void</td>
  </tr>
 </tbody>
</table>
```

<h4>1. Basic data types in C:</h4>

<h4>1.1. Integer data type:</h4>

Integer data type allows a variable to store numeric values.

“int” keyword is used to refer integer data type.

The storage size of int data type is 2 or 4 or 8 byte.

It varies depend upon the processor in the CPU that we use. If we are using 16 bit processor, 2 byte (16 bit) of memory will be allocated for int data type.

Like wise, 4 byte (32 bit) of memory for 32 bit processor and 8 byte (64 bit) of memory for 64 bit processor is allocated for int datatype.

int (2 byte) can store values from -32,768 to +32,767

int (4 byte) can store values from -2,147,483,648 to +2,147,483,647.

If you want to use the integer value that crosses the above limit, you can go for “long int” and “long long int” for which the limits are very high.

Note:

We can’t store decimal values using int data type.

If we use int data type to store decimal values, decimal values will be truncated and we will get only whole number.

In this case, float data type can be used to store decimal values in a variable.

<h4>1.2. Character data type:</h4>

Character data type allows a variable to store only one character.

Storage size of character data type is 1. We can store only one character using character data type.

“char” keyword is used to refer character data type.

For example, ‘A’ can be stored using char datatype. You can’t store more than one character using char data type.

Please refer C – Strings topic to know how to store more than one characters in a variable.

<h4>1.3. Floating point data type:</h4>

Floating point data type consists of 2 types. They are,

float

double

<h4>1. float:</h4>

```
<ul>
 <li>Float data type allows a variable to store decimal values.</li>
 <li>Storage size of float data type is 4. This also varies depend upon the processor in the CPU as &ldquo;int&rdquo; data type.</li>
 <li>We can use up-to 6 digits after decimal using float data type.</li>
 <li>For example, 10.456789 can be stored in a variable using float data type.</li>
</ul>
<h4><strong>2. double:</strong></h4>
<ul>
 <li>Double data type is also same as float data type which allows up-to 10 digits after decimal.</li>
 <li>The range for double datatype is from 1E–37 to 1E+37.</li>
</ul>
<h4><strong>1.3.1. sizeof() function in C:</strong></h4>
<p>sizeof() function is used to find the memory space allocated for each C data types.</p>
<h2>Example Program:</h2>
<div class="program">#include &lt;stdio.h&gt;
#include &lt;limits.h&gt;
int main()
{
        int a;
        char b;
        float c;
        double d;
        printf(&ldquo;Storage size for int data type:%d \n&rdquo;,sizeof(a));
        printf(&ldquo;Storage size for char data type:%d \n&rdquo;,sizeof(b));
        printf(&ldquo;Storage size for float data type:%d \n&rdquo;,sizeof(c));
        printf(&ldquo;Storage size for double data type:%d\n&rdquo;,sizeof(d));
        return 0;
}</div>
<h2>Output:</h2>
<div class="output">Storage size for int data type:4
Storage size for char data type:1
Storage size for float data type:4
Storage size for double data type:8
</div>
<h4><strong>1.3.2. Modifiers in C:</strong></h4>
<ul>
```

The amount of memory space to be allocated for a variable is derived by modifiers.

Modifiers are prefixed with basic data types to modify (either increase or decrease) the amount of storage space allocated to a variable.

For example, storage space for int data type is 4 byte for 32 bit processor. We can increase the range by using long int which is 8 byte. We can decrease the range by using short int which is 2 byte.

There are 5 modifiers available in C language. They are,

short

long

signed

unsigned

long long

Below table gives the detail about the storage size of each C basic data type in 16 bit processor.

Please keep in mind that storage size and range for int and float datatype will vary depend on the CPU processor (8,16, 32 and 64 bit)

<table width="800" border="1">

<tbody>

<tr>

<td width="36">S.No</td>

<td width="181">C Data types</td>

<td width="107">storage Size</td>

<td width="448">Range</td>

</tr>

<tr>

<td>1</td>

<td>char</td>

<td>1</td>

<td>–127 to 127</td>

</tr>

<tr>

<td>2</td>

<td>int</td>

```
 <td>2</td>
 <td>–32,767 to 32,767</td>
</tr>
<tr>
 <td>3</td>
 <td>float</td>
 <td>4</td>
 <td>1E–37 to 1E+37 with six digits of precision</td>
</tr>
<tr>
 <td>4</td>
 <td>double</td>
 <td>8</td>
 <td>1E–37 to 1E+37 with ten digits of precision</td>
</tr>
<tr>
 <td>5</td>
 <td>long double</td>
 <td>10</td>
 <td>1E–37 to 1E+37 with ten digits of precision</td>
</tr>
<tr>
 <td>6</td>
 <td>long int</td>
 <td>4</td>
 <td>–2,147,483,647 to 2,147,483,647</td>
</tr>
<tr>
 <td>7</td>
 <td>short int</td>
 <td>2</td>
 <td>–32,767 to 32,767</td>
</tr>
<tr>
 <td>8</td>
 <td>unsigned short int</td>
```

```
    <td>2</td>
    <td>0 to 65,535</td>
   </tr>
   <tr>
    <td>9</td>
    <td>signed short int</td>
    <td>2</td>
    <td>–32,767 to 32,767</td>
   </tr>
   <tr>
    <td>10</td>
    <td>long long int</td>
    <td>8</td>
    <td>–(2power(63) –1) to 2(power)63 –1</td>
   </tr>
   <tr>
    <td>11</td>
    <td>signed long int</td>
    <td>4</td>
    <td>–2,147,483,647 to 2,147,483,647</td>
   </tr>
   <tr>
    <td>12</td>
    <td>unsigned long int</td>
    <td>4</td>
    <td>0 to 4,294,967,295</td>
   </tr>
   <tr>
    <td>13</td>
    <td>unsigned long long int</td>
    <td>8</td>
    <td>2(power)64 –1</td>
   </tr>
  </tbody>
</table>
<h4><strong><strong>2. Enumeration data type in C:</strong></strong></h4>
```

```
  <ul>
   <li>Enumeration data type consists of named integer constants as a list.</li>
   <li>It start with 0 (zero) by default and value is incremented by 1 for the sequential identifiers in the list.</li>
  </ul>
  <ul>
   <li>Enum syntax in C:</li>
  </ul>
  <div class="syntax">enum identifier [optional{ enumerator-list }];</div>
  <h4><strong>C – enum example program:</strong></h4>
  <div class="program">#include &lt;stdio.h&gt;
int main()
{
        enum MONTH { Jan = 0, Feb, Mar };
        enum MONTH month = Mar;
        if(month == 0)
                printf(&ldquo;Value of Jan&rdquo;);
        else if(month == 1)
                printf(&ldquo;Month is Feb&rdquo;);
        if(month == 2)
                printf(&ldquo;Month is Mar&rdquo;);
}</div>
  <h2>Output:</h2>
  <div class="output">Month is March</div>
  <h4><strong>3. Derived data type in C:</strong></h4>
  <ul>
   <li>Array, pointer, structure and union are called derived data type in C language.</li>
   <li>To know more about derived data types, please visit &ldquo;<a title="Array" href="arrays.html">C – Array</a>&ldquo; , &ldquo;<a title="Pointer" href="Pointers.html">C – Pointer</a>&rdquo; and &ldquo;<a title="Structures" href="Structures.html">C – Structure</a>&rdquo; topics in this tutorial.</li>
  </ul>
  <h4><strong>4. Void data type in C:</strong></h4>
  <ul>
   <li>Void is an empty data type that has no value.</li>
   <li>This can be used in functions and pointers.</li>
   <li>Please visit &ldquo;<a title="Function" href="Functions.html">C – Function</a>&rdquo; topic to know how to use void data type in function with simple call by value and call by reference example programs.</li>
  </ul>
 </div>
```

```
<div id="Footer">
            <ul>
            <li><a href="PrintfAndScanfFunctions.html"><<</a></li>
    <li><a href="PrintfAndScanfFunctions.html">1</a></li>
    <li><a href="DataTypes.html">2</a></li>
    <li><a href="Constants.html">3</a></li>
    <li><a href="Variables.html">4</a></li>
    <li><a href="OperatorsAndExpressions.html">5</a></li>
    <li><a href="DecisionControlStatements.html">6</a></li>
    <li><a href="LoopControlStatements.html">7</a></li>
    <li><a href="CaseControlStatements.html">8</a></li>
    <li><a href="Arrays.html">9</a></li>
    <li><a href="String.html">10</a></li>
    <li><a href="Pointers.html">11</a></li>
    <li><a href="Functions.html">12</a></li>
    <li><a href="Structures.html">13</a></li>
    <li><a href="Constants.html">>></a></li>
  </ul>
 </div>
</div>
</body>
</html>
```

Constants.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Constants</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content">

            <h1 style="text-align:center;">Constant</h1>

            <ul>
```

```
            <li>C Constants are also like normal variables. But, only difference is, their values can not be modified by the program once they are defined.</li>

            <li>Constants refer to fixed values. They are also called as literals</li>

            <li>Constants may be belonging to any of the data type.</li>

  </ul>

            <h4>Syntax:</h4>

           <p class="syntax">const data_type variable_name;

           (or)

const data_type *variable_name;</p>

           <h4><strong>Types of C constant:</strong></h4>

<ol type="1">

            <li>Integer constants</li>

            <li>Real or Floating point constants</li>

            <li>Octal &amp;amp; Hexadecimal constants</li>

            <li>Character constants</li>

            <li>String constants</li>

            <li>Backslash character constants</li>

  </ol>

           <table width="800" border="1" cellpadding="2">

            <tbody>

              <tr>

                <td width="34" style="text-align: center">S.no</td>

                <td width="262" style="text-align: center"><div>Constant type</div></td>

                <td width="133" style="text-align: center"><div>data type</div></td>

                <td width="335" style="text-align: center"><div>Example</div></td>

    </tr>

              <tr>

                <td style="text-align: center">1</td>

                <td>Integer constants</td>

                <td>int<br>

                  unsigned int<br>

                  long int<br>

                  long long int</td>

                <td>53, 762, -478 etc <br>

                  5000u, 1000U etc<br>

                  483,647<br>

                  2,147,483,680</td>
```

```
  </tr>
            <tr>
             <td style="text-align: center">2</td>
             <td>Real or Floating point constants</td>
             <td>float<br>
              doule</td>
             <td>10.456789<br>
              600.123456789</td>
  </tr>
            <tr>
             <td style="text-align: center">3</td>
             <td>Octal constant</td>
             <td>int</td>
             <td>013        /* starts with 0  */</td>
  </tr>
            <tr>
             <td style="text-align: center">4</td>
             <td>Hexadecimal constant</td>
             <td>int</td>
             <td>0×90       /* starts with 0x */</td>
  </tr>
            <tr>
             <td style="text-align: center">5</td>
             <td>character constants</td>
             <td><div>char</div></td>
             <td>&lsquo;A&rsquo;  ,  &lsquo;B&rsquo;,    &lsquo;C&rsquo;</td>
  </tr>
            <tr>
             <td style="text-align: center">6</td>
             <td>string constants</td>
             <td><div>char</div></td>
             <td>&ldquo;ABCD&rdquo;  ,  &ldquo;Hai&rdquo;</td>
  </tr>
 </tbody>
</table>
        <h4><strong>Rules for constructing C constant:</strong></h4>
```

```
<h4><strong>1. Integer Constants in C:</strong></h4>
<ul>
 <li>An integer constant must have at least one digit.</li>
 <li>It must not have a decimal point.</li>
 <li>It can either be positive or negative.</li>
 <li>No commas or blanks are allowed within an integer constant.</li>
 <li>If no sign precedes an integer constant, it is assumed to be positive.</li>
 <li>The allowable range for integer constants is -32768 to 32767.</li>
</ul>
<h4><strong>2. Real constants in C:</strong></h4>
<ul>
 <li>A real constant must have at least one digit</li>
 <li>It must have a decimal point</li>
 <li>It could be either positive or negative</li>
 <li>If no sign precedes an integer constant, it is assumed to be positive.</li>
 <li>No commas or blanks are allowed within a real constant.</li>
</ul>
<h4><strong>3. Character and string constants in C:</strong></h4>
<ul>
 <li>A character constant is a single alphabet, a single digit or a single special symbol enclosed within single quotes.</li>
 <li>The maximum length of a character constant is 1 character.</li>
 <li>String constants are  enclosed within double quotes.</li>
</ul>
<h4><strong>4. Backslash Character Constants in C:</strong></h4>
<ul>
 <li>There are some characters which have special meaning in C language.</li>
 <li>They should be preceded by backslash symbol to make use of special function of them.</li>
 <li>Given below is the list of special characters and their purpose.</li>
</ul>
<div>
 <table width="500" border="2" cellpadding="2">
  <tbody>
   <tr style="font-weight: bold; text-align: center;">
    <td>Backslash_character</td>
    <td>Meaning</td>
  </tr>
```

```
            <tr>
             <td style="text-align: center">\b</td>
             <td style="text-align: left">Backspace</td>
</tr>
            <tr>
             <td style="text-align: center">\f</td>
             <td style="text-align: left">Form feed</td>
</tr>
            <tr>
             <td style="text-align: center">\n</td>
             <td style="text-align: left">New line</td>
</tr>
            <tr>
             <td style="text-align: center">\r</td>
             <td style="text-align: left">Carriage return</td>
</tr>
            <tr>
             <td style="text-align: center">\t</td>
             <td style="text-align: left">Horizontal tab</td>
</tr>
            <tr>
             <td style="text-align: center">\&rdquo;</td>
             <td style="text-align: left">Double quote</td>
</tr>
            <tr>
             <td style="text-align: center">\&rsquo;</td>
             <td style="text-align: left">Single quote</td>
</tr>
            <tr>
             <td style="text-align: center">\\</td>
             <td style="text-align: left">Backslash</td>
</tr>
            <tr>
             <td style="text-align: center">\v</td>
             <td style="text-align: left">Vertical tab</td>
</tr>
```

```
                <tr>
                 <td style="text-align: center">\a</td>
                 <td style="text-align: left">Alert or bell</td>
      </tr>
                <tr>
                 <td style="text-align: center">\?</td>
                 <td style="text-align: left">Question mark</td>
      </tr>
                <tr>
                 <td style="text-align: center">\N</td>
                 <td style="text-align: left">Octal constant (N is an octal constant)</td>
      </tr>
                <tr>
                 <td style="text-align: center">\XN</td>
                 <td style="text-align: left">Hexadecimal constant (N – hex.dcml cnst)</td>
      </tr>
     </tbody>
   </table>
</div>
            <div>
              <h4><strong>How to use constants in a C program?</strong></h4>
              <ul>
                <li>We can define constants in a C program in the following ways.</li>
   </ul>
              <ol>
                <li>By &amp;ldquo;const&amp;rdquo; keyword</li>
                <li>By &amp;ldquo;#define&amp;rdquo; preprocessor directive</li>
   </ol>
              <ul>
                <li>Please note that when you try to change constant values after defining in C program, it will through error.</li>
   </ul>
              <h4><strong>1. Example program using const keyword in C:</strong></h4>
  </div>
            <div class="program">#inlcude&amp;lt;stdio.h&amp;gt;
void main()
{
```

```
        const int height = 100;                         /*int constant*/
        const float number = 3.14;                              /*Real constant*/
        const char letter = 'A';                        /*char constant*/
        const char letter_sequence[10] = "ABC";         /*string constant*/
        const char backslash_char = '\?';       /*special char cnst*/
        printf("value of height :%d \n", height );
        printf("value of number : %f \n", number );
        printf("value of letter : %c \n", letter );
        printf("value of letter_sequence : %s \n", letter_sequence);
        printf("value of backslash_char : %c \n", backslash_char);
}</div>
  <h4><strong>Output:</strong></h4>
  <div class="output">value of height : 100
value of number : 3.140000
value of letter : A
value of letter_sequence : ABC
value of backslash_char : ? </div>
  <h4><strong>2. Example program using #define preprocessor directive in C:</strong></h4>
  <div class="program">#define height 100
#define number 3.14
#define letter 'A'
#define letter_sequence "ABC"
#define backslash_char '\?'
void main()
{
        printf("value of height : %d \n", height );
        printf("value of number : %f \n", number );
        printf("value of letter : %c \n", letter );
        printf("value of letter_sequence : %s \n",letter_sequence);
        printf("value of backslash_char : %c \n",backslash_char);
}</div>
  <h4><strong>Output:</strong></h4>
<div class="output">value of height : 100
value of number : 3.140000
value of letter : A
value of letter_sequence : ABC
```

```
value of backslash_char : ?</div>
 </div>

 <div id="Footer">
            <ul>
            <li><a href="DataTypes.html"><<</a></li>
     <li><a href="PrintfAndScanfFunctions.html">1</a></li>
     <li><a href="DataTypes.html">2</a></li>
     <li><a href="Constants.html">3</a></li>
     <li><a href="Variables.html">4</a></li>
     <li><a href="OperatorsAndExpressions.html">5</a></li>
     <li><a href="DecisionControlStatements.html">6</a></li>
     <li><a href="LoopControlStatements.html">7</a></li>
     <li><a href="CaseControlStatements.html">8</a></li>
     <li><a href="Arrays.html">9</a></li>
     <li><a href="String.html">10</a></li>
     <li><a href="Pointers.html">11</a></li>
     <li><a href="Functions.html">12</a></li>
     <li><a href="Structures.html">13</a></li>
     <li><a href="Variables.html">></a></li>
   </ul>
 </div>
</div>
</body>
</html>
```

Variables.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Variables</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content">

 <h1 style="text-align:center;">Variable</h1>

 <ul>
```

C variable is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable.

The value of the C variable may get change in the program.

C variable might be belonging to any of the data type like int, float, char etc.

<h4>Rules for naming C variable:</h4>

Variable name must begin with letter or underscore.

Variables are case sensitive

They can be constructed with digits, letters.

No special symbols are allowed other than underscore.

sum, height, _value are some examples for variable name

<h4>Declaring & initializing C variable:</h4>

Variables should be declared in the C program before to use.

Memory space is not allocated for a variable while declaration. It happens only on variable definition.

Variable initialization means assigning a value to the variable.

<table width="900" border="2" cellpadding="3">

<tbody>

<tr>

<td width="40" style="text-align: center"> S.No </td>

<td width="182" style="text-align: center"> Type </td>

<td width="271" style="text-align: center"> Syntax </td>

<td width="361" style="text-align: center"> Example </td>

</tr>

<tr>

<td style="text-align: center"> 1 </td>

<td> Variable declaration </td>

<td> data_type variable_name; </td>

<td> int x, y, z; char flat, ch; </td>

</tr>

<tr>

<td style="text-align: center"> 2 </td>

<td> Variable initialization </td>

<td> data_type variable_name = value; </td>

```
      <td><div>int x = 50, y = 30; char flag = &lsquo;x&rsquo;, ch=&rsquo;l&rsquo;;</div></td>
    </tr>
  </tbody>
 </table>
 <h4><strong>There are three types of variables in C program They are,</strong></h4>
 <ol>
  <li>Local variable</li>
  <li>Global variable</li>
  <li>Environment variable</li>
 </ol>
 <h4><strong>1. Example program for local variable in C:</strong></h4>
 <ul>
  <li>The scope of local variables will be within the function only.</li>
  <li>These variables are declared within the function and can&rsquo;t be accessed outside the function.</li>
  <li>In the below example, m and n variables are having scope within the main function only. These are not visible to test function.</li>
  <li>Like wise, a and b variables are having scope within the test function only. These are not visible to main function.</li>
 </ul>
 <div class="program">#include< stdio.h>
void test();
int main()
{
        int m = 22, n = 44;      // m, n are local variables of main function
        /*m and n variables are having scope within this main function only. These are not visible to test funtion.*/
        /* If you try to access a and b in this function, you will get ‘a’ undeclared and ‘b’ undeclared error */
        printf(“\n values : m = %d and n = %d”, m, n);
        test();
}
void test()
{
        int a = 50, b = 80;    // a, b are local variables of test function
        /*a and b variables are having scope within this test function only. These are not visible to main function.*/
        /* If you try to access m and n in this function, you will get ‘m’ undeclared and ‘n’ undeclared error */
        printf(“\nvalues : a = %d and b = %d”, a, b);
}</div>
<h4>Output:</h4>
<div class="output">values : m = 22 and n = 44
```

values : a = 50 and b = 80

<h4>2. Example program for global variable in C:</h4>

The scope of global variables will be throughout the program. These variables can be accessed from anywhere in the program.

This variable is defined outside the main function. So that, this variable is visible to main function and all other sub functions.

#include< stdio.h>

void test();

int m = 22, n = 44;

int a = 50, b = 80;

int main()

{

printf("All variables are accessed from main function");

printf("\nvalues: m=%d:n=%d:a=%d:b=%d", m,n,a,b);

test();

}

void test()

{

printf("\n\ng All variables are accessed from test function");

printf("\n values: m=%d:n=%d:a=%d:b=%d", m,n,a,b);

}

<h4>Output:</h4>

All variables are accessed from main function

values : m = 22 : n = 44 : a = 50 : b = 80

All variables are accessed from test function

values : m = 22 : n = 44 : a = 50 : b = 80

<h4>3. Environment variables in C:</h4>

Environment variable is a variable that will be available for all C applications and C programs.

We can access these variables from anywhere in a C program without declaring and initializing in an application or C program.

The inbuilt functions which are used to access, modify and set these environment variables are called environment functions.

There are 3 functions which are used to access, modify and assign an environment variable in C. They are,

1. setenv()

2. getenv()

3. putenv()

<h4>Example program for getenv() function in C:</h4>

This function gets the current value of the environment variable. Let us assume that environment variable DIR is assigned to “/usr/bin/test/”.

```
<div class="program">#include< stdio.h>

#include< stdlib.h>

int main()

{

        printf("Directory = %s\n",getenv("DIR"));

        return 0;

}</div>
```

<h4>Output:</h4>

/usr/bin/test/

<h4> Example program for setenv() function in C:</h4>

This function sets the value for environment variable. Let us assume that environment variable “FILE” is to be assigned “/usr/bin/example.c”

```
<div class="program">#include< stdio.h>

#include< stdlib.h>

int main()

{

        setenv("FILE", "/usr/bin/example.c",50);

        printf("File = %s\n", getenv("FILE"));

        return 0;

}</div>
```

<h4>Output:</h4>

File = /usr/bin/example.c

<h4>Example program for putenv() function in C:</h4>

This function modifies the value for environment variable. Below example program shows that how to modify an existing environment variable value.

```
<div class="program">#include< stdio.h>

#include< stdlib.h>

int main()

{

        setenv("DIR", "/usr/bin/example/",50);

        printf("Directory name before modifying = " \"%s\n", getenv("DIR"));

        putenv("DIR=/usr/home/");
```

```
        printf("Directory name after modifying = " \"%s\n", getenv("DIR"));
        return 0;
}
</div>
<h4>Output:</h4>
<div class="output">Directory name before modifying = /usr/bin/example/
Directory name after modifying = /usr/home/</div>
<h4><strong>Difference between variable declaration &amp;amp; definition in C:</strong></h4>
<table width="900" border="1" cellpadding="3">
 <tbody>
  <tr>
   <td><div><strong>S.no </strong></div></td>
   <td><div><strong>Variable declaration</strong></div></td>
   <td><div><strong>Variable definition</strong></div></td>
  </tr>
  <tr>
   <td><div>1</div></td>
   <td><div>Declaration tells the compiler about data type and size of the variable.</div></td>
   <td><div>Definition allocates memory for the variable.</div></td>
  </tr>
  <tr>
   <td><div>2</div></td>
   <td><div>Variable can be declared many times in a program.</div></td>
   <td><div>It can happen only one time for a variable in a program.</div></td>
  </tr>
  <tr>
   <td><div>3</div></td>
   <td><div>The assignment of properties and identification to a variable.</div></td>
   <td><div>Assignments of storage space to a variable.</div></td>
  </tr> </tbody></table> </div>
 <div id="Footer">
        <ul>
        <li><a href="Constants.html"><</a></li>
    <li><a href="PrintfAndScanfFunctions.html">1</a></li>
    <li><a href="DataTypes.html">2</a></li>
    <li><a href="Constants.html">3</a></li>
```

```
      <li><a href="Variables.html">4</a></li>
      <li><a href="OperatorsAndExpressions.html">5</a></li>
      <li><a href="DecisionControlStatements.html">6</a></li>
      <li><a href="LoopControlStatements.html">7</a></li>
      <li><a href="CaseControlStatements.html">8</a></li>
      <li><a href="Arrays.html">9</a></li>
      <li><a href="String.html">10</a></li>
      <li><a href="Pointers.html">11</a></li>
      <li><a href="Functions.html">12</a></li>
      <li><a href="Structures.html">13</a></li>
      <li><a href="OperatorsAndExpressions.html">>></a></li>
    </ul>
</div>
</div>
</body>
</html>
```

OperatorsAndExpressions.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Operators And Expressions</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content"><h1 style="text-align:center;">Operators And Expressions</h1>

   <ul>

    <li>The symbols which are used to perform logical and mathematical operations in a C program are called C operators.</li>
```

```
  <li>These C operators join individual constants and variables to form expressions.</li>
  <li>Operators, functions, constants and variables are combined together to form expressions.</li>
  <li>Consider the expression A + B * 5. where, +, * are operators, A, B  are variables, 5 is constant and A + B * 5 is an expression.</li>
 </ul>
 <h4><strong>Types of C operators:</strong></h4>
 <p>C language offers many types of operators. They are,</p>
 <ol start="1" type="1">
  <li>Arithmetic operators</li>
  <li>Assignment operators</li>
  <li>Relational operators</li>
  <li>Logical operators</li>
  <li>Bit wise operators</li>
  <li>Conditional operators (ternary operators)</li>
  <li>Increment/decrement operators</li>
  <li>Special operators</li>
 </ol>
 <table width="900" border="1" cellpadding="4">
  <tbody>
   <tr>
    <td width="38" style="text-align: center"><div><strong>S.no</strong></div></td>
    <td width="200" style="text-align: center"><div><strong>Types of Operators</strong></div></td>
    <td width="622" style="text-align: center"><div><strong>Description</strong></div></td>
   </tr>
   <tr>
    <td style="text-align: left"><div>1</div></td>
    <td style="text-align: left"><strong>Arithmetic_operators</strong></td>
    <td style="text-align: justify">These are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus</td>
   </tr>
   <tr>
    <td style="text-align: left"><div>2</div></td>
    <td style="text-align: left"><strong>Assignment_operators</strong></td>
    <td style="text-align: justify">These are used to assign the values for the variables in C programs.</td>
   </tr>
   <tr>
    <td style="text-align: left"><div>3</div></td>
    <td style="text-align: left"><strong>Relational operators</strong></td>
```

<td style="text-align: justify">These operators are used to compare the value of two variables.</td>

</tr>

<tr>

<td style="text-align: left"> 4 </td>

<td style="text-align: left">Logical operators</td>

<td style="text-align: justify">These operators are used to perform logical operations on the given two variables.</td>

</tr> <tr>

<td style="text-align: left"> 5 </td>

<td style="text-align: left">Bit wise operators</td>

<td style="text-align: justify">These operators are used to perform bit operations on given two variables.</td>

</tr> <tr>

<td style="text-align: left"> 6 </td>

<td style="text-align: left">Conditional (ternary) operators</td>

<td style="text-align: justify">Conditional operators return one value if condition is true and returns another value is condition is false.</td>

</tr> <tr>

<td style="text-align: left"> 7 </td>

<td style="text-align: left">Increment/decrement operators</td>

<td style="text-align: justify">These operators are used to either increase or decrease the value of the variable by one.</td>

</tr> <tr>

<td style="text-align: left"> 8 </td>

<td style="text-align: left">Special operators</td>

<td style="text-align: justify">&, *, sizeof( ) and ternary operators.</td>

</tr> </tbody> </table> <h4>1. Arithmetic Operators in C:</h4>

C Arithmetic operators are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus in C programs.

<table width="500" border="1" cellpadding="3">

<tbody> <tr>

<td style="text-align: center"> S.no </td>

<td style="text-align: center"> Arithmetic Operators </td>

<td style="text-align: center"> Operation </td>

<td style="text-align: center"> Example </td>

</tr> <tr>

<td style="text-align: center"> 1 </td>

<td style="text-align: center"> + </td>

<td style="text-align: center"> Addition </td>

<td style="text-align: center"> A+B </td>

</tr> <tr>

<td style="text-align: center"> 2 </td>

<td style="text-align: center"> - </td>

<td style="text-align: center"> Subtraction </td>

<td style="text-align: center"> A-B </td>

</tr> <tr>

<td style="text-align: center"> 3 </td>

<td style="text-align: center"> * </td>

<td style="text-align: center"> multiplication </td>

<td style="text-align: center"> A*B </td>

</tr> <tr>

<td style="text-align: center"> 4 </td>

<td style="text-align: center"> / </td>

<td style="text-align: center"> Division </td>

<td style="text-align: center"> A/B </td>

</tr> <tr>

<td style="text-align: center"> 5 </td>

<td style="text-align: center"> % </td>

<td style="text-align: center"> Modulus </td>

<td style="text-align: center"> A%B </td>

</tr> </tbody> </table> <h4>Example program for C arithmetic operators:</h4>

In this example program, two values “40” and “20” are used to perform arithmetic operations such as addition, subtraction, multiplication, division, modulus and output is displayed for each operation.

#include< stdio.h>

```
int main()

{

int a=40,b=20, add,sub,mul,div,mod;

add = a+b;

sub = a-b;

mul = a*b;

div = a/b;

mod = a%b;

printf(“Addition of a, b is : %d\n”, add);
```

printf(“Subtraction of a, b is : %d\n”, sub);

printf(“Multiplication of a, b is : %d\n”, mul);

printf(“Division of a, b is : %d\n”, div);

printf(“Modulus of a, b is : %d\n”, mod);

}

<h4>Output:</h4>

Addition of a, b is : 60

Subtraction of a, b is : 20

Multiplication of a, b is : 800

Division of a, b is : 2

Modulus of a, b is : 0

<h4>2. Assignment operators in C:</h4>

In C programs, values for the variables are assigned using assignment operators.

For example, if the value “10” is to be assigned for the variable “sum”, it can be assigned as “sum = 10;”

Other assignment operators in C language are given below.

<table width="900" border="1" cellpadding="3">

<tbody> <tr>

<td height="33" colspan="2" style="text-align: center"> Operators </td>

<td width="125" style="text-align: center"> Example </td>

<td width="345" style="text-align: center"> Explanation </td>

</tr> <tr>

<td width="287" style="text-align: center"> Simple assignment operator </td>

<td width="99" style="text-align: center"> = </td>

<td style="text-align: center"> sum = 10 </td>

<td style="text-align: center"> 10 is assigned to variable sum </td>

</tr> <tr>

<td rowspan="7" style="text-align: center"> Compound assignment operators </td>

<td style="text-align: center"> += </td>

<td style="text-align: center"> sum += 10 </td>

<td style="text-align: center"> This is same as sum = sum + 10 </td>

</tr> <tr>

<td style="text-align: center"> -= </td>

<td style="text-align: center"> sum -= 10 </td>

<td style="text-align: center"> This is same as sum = sum – 10 </td>

</tr> <tr>

<td style="text-align: center"> *= </td>

<td style="text-align: center"> sum *= 10 </td>

<td style="text-align: center"> This is same as sum = sum * 10 </td>

</tr> <tr>

<td style="text-align: center"> /+ </td>

<td style="text-align: center"> sum /= 10 </td>

<td style="text-align: center"> This is same as sum = sum / 10 </td>

</tr> <tr>

<td style="text-align: center"> %= </td>

<td style="text-align: center"> sum %= 10 </td>

<td style="text-align: center"> This is same as sum = sum % 10 </td>

</tr> <tr>

<td style="text-align: center"> &= </td>

<td style="text-align: center"> sum&=10 </td>

<td style="text-align: center"> This is same as sum = sum & 10 </td>

</tr> <tr>

<td style="text-align: center"> ^= </td>

<td style="text-align: center"> sum ^= 10 </td>

<td style="text-align: center"> This is same as sum = sum ^ 10 </td>

</tr> </tbody></table><h4>Example program for C assignment operators:</h4>

In this program, values from 0 – 9 are summed up and total “45” is displayed as output.

Assignment operators such as “=” and “+=” are used in this program to assign the values and to sum up the values.

#include< stdio.h>

int main()

{

int Total=0,i;

for(i = 0;i < 10;i++)

{

Total+=i; // This is same as Total = Toatal+i

}

printf(“Total = %d”, Total);

}

```
<h4>Output:</h4>

<div class="output">Total = 45</div><h4><strong>3. Relational operators in C:</strong></h4>

<div>  <ul>

   <li>Relational operators are used to find the relation between two variables. i.e. to compare the values of two variables in a C program.</li>

 </ul>

</div>

<div>

 <table width="500" border="1" cellpadding="3">

  <tbody>

   <tr>

    <td width="38" style="text-align: center"><div><span data-mce-mark="1"><strong>S.no</strong></span></div></td>

    <td width="88" style="text-align: center"><div><span data-mce-mark="1"><strong>Operators</strong></span></div></td>

    <td width="91" style="text-align: center"><div><span data-mce-mark="1"><strong>Example</strong></span></div></td>

    <td width="239" style="text-align: center"><div><span data-mce-mark="1"><strong>Description</strong></span></div></td>

   </tr>    <tr>

    <td style="text-align: center"><div>1</div></td>

    <td style="text-align: center"><div>&gt;</div></td>

    <td style="text-align: center"><div>x &gt; y</div></td>

    <td style="text-align: center"><div>x is greater than y</div></td>

   </tr>    <tr>

    <td style="text-align: center"><div>2</div></td>

    <td style="text-align: center"><div>&lt;</div></td>

    <td style="text-align: center"><div>x &lt; y</div></td>

    <td style="text-align: center"><div>x is less than y</div></td>

   </tr>    <tr>

    <td style="text-align: center"><div>3</div></td>

    <td style="text-align: center"><div>&gt;=</div></td>

    <td style="text-align: center"><div>x &gt;= y</div></td>

    <td style="text-align: center"><div>x is greater than or equal to y</div></td>

   </tr>    <tr>

    <td style="text-align: center"><div>4</div></td>

    <td style="text-align: center"><div>&lt;=</div></td>

    <td style="text-align: center"><div>x &lt;= y</div></td>

    <td style="text-align: center"><div>x is less than or equal to y</div></td>

   </tr>    <tr>

    <td style="text-align: center"><div>5</div></td>
```

```
      <td style="text-align: center"><div>==</div></td>

      <td style="text-align: center"><div>x == y</div></td>

      <td style="text-align: center"><div>x is equal to y</div></td>

    </tr>     <tr>

      <td style="text-align: center"><div>6</div></td>

      <td style="text-align: center"><div>!=</div></td>

      <td style="text-align: center"><div>x != y</div></td>

      <td style="text-align: center"><div>x is not equal to y</div></td>

    </tr>   </tbody>  </table></div>

<h4><strong>Example program for relational operators in C:</strong></h4>

<ul>

  <li>In this program, relational operator (==) is used to compare 2 values whether they are equal are not.</li>

  <li>If both values are equal, output is displayed as &rdquo; values are equal&rdquo;. Else, output is displayed as &ldquo;values are not equal&rdquo;.</li>

  <li>Note : double equal sign (==) should be used to compare 2 values. We should not single equal sign (=).</li>

</ul>

<div class="program">#include< stdio.h>

int main()

{

int m=40,n=20;

if (m == n)

{

printf("m and n are equal");

}

else

{

printf("m and n are not equal");

}

}</div>

<h4>Output:</h4>

<div class="output">m and n are not equal</div>

<h4><strong>4. Logical operators in C:</strong></h4>

<div>  <ul>

    <li>These operators are used to perform logical operations on the given expressions.</li>

    <li>There are 3 logical operators in C language. They are, logical AND (&amp;amp;amp;), logical OR (||) and logical NOT (!).</li>

  </ul></div>

<div>
```

<table width="900" border="1" cellpadding="3"> <tbody> <tr>

<td width="38" style="text-align: center"> S.no </td>

<td width="88" style="text-align: center"> Operators </td>

<td width="90" style="text-align: center"> Name </td>

<td width="144" style="text-align: center"> Example </td>

<td width="486" style="text-align: center"> Description </td>

</tr> <tr>

<td style="text-align: center"> 1 </td>

<td style="text-align: center"> && </td>

<td style="text-align: center"> logical AND </td>

<td style="text-align: center"> (x>5)&&(y<5) </td>

<td style="text-align: justify"> It returns true when both conditions are true </td>

</tr> <tr>

<td style="text-align: center"> 2 </td>

<td style="text-align: center"> || </td>

<td style="text-align: center"> logical OR </td>

<td style="text-align: center"> (x>=10)||(y>=10) </td>

<td style="text-align: justify"> It returns true when at-least one of the condition is true </td>

</tr> <tr>

<td style="text-align: center"> 3 </td>

<td style="text-align: center"> ! </td>

<td style="text-align: center"> logical NOT </td>

<td style="text-align: center"> !((x>5)&&(y<5)) </td>

<td style="text-align: justify"> It reverses the state of the operand “((x>5) && (y<5))”

If “((x>5) && (y<5))” is true, logical NOT operator makes it false </td>

</tr> </tbody> </table> <h4> Example program for logical operators in C:</h4>

#include< stdio.h>

int main()

{

int m=40,n=20;

int o=20,p=30;

if (m>n && m !=0)

{

printf("&& Operator : Both conditions are true\n");

}

if (o>p || p!=20)

{

printf(“|| Operator : Only one condition is true\n”);

}

if (!(m>n && m !=0))

{

printf(“! Operator : Both conditions are true\n”);

}

else

{

printf(“! Operator : Both conditions are true. But, status is inverted as false\n”);

}

}

<h4>Output:</h4>

&& Operator : Both conditions are true

|| Operator : Only one condition is true

! Operator : Both conditions are true. But, status is inverted as false

In this program, operators (&&,|| and !) are used to perform logical operations on the given expressions.

&& operator – “if clause” becomes true only when both conditions (m>n and m! =0) is true. Else, it becomes false.

|| Operator – “if clause” becomes true when any one of the condition (o>p || p!=20) is true. It becomes false when none of the condition is true.

! Operator – It is used to reverses the state of the operand.

If the conditions (m>n && m!=0) is true, true (1) is returned. This value is inverted by “!” operator.

So, “! (m>n and m!=0)” returns false (0).

<h4>5. Bit wise operators in C:</h4>

These operators are used to perform bit operations. Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.

Bit wise operators in C language are & (bitwise AND), | (bitwise OR), ~ (bitwise OR), ^ (XOR), << (left shift) and >> (right shift).

<h4>Truth table for bit wise operation Bit wise operators</h4>

<table width="800" border="1" cellpadding="2"> <tbody> <tr>

<td width="61" style="font-weight: bold; text-align: center;"> x </td>

<td width="60" style="font-weight: bold; text-align: center;"> y </td>

<td width="63" style="font-weight: bold; text-align: center;"> x|y </td>

<td width="64" style="font-weight: bold; text-align: center;"> x & y </td>

```
  <td width="66" style="font-weight: bold; text-align: center;"><div>x ^ y</div></td>
  <td width="17" style="text-align: center"></td>
  <td width="203" style="text-align: center; font-weight: bold;"><div>Operator_symbol</div></td>
  <td width="198" style="text-align: center; font-weight: bold;"><div>Operator_name</div></td>
 </tr>  <tr>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"></td>
  <td style="text-align: center"><div>&amp;amp;</div></td>
  <td style="text-align: center"><div>Bitwise_AND</div></td>
 </tr>  <tr>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"></td>
  <td style="text-align: center"><div>|</div></td>
  <td style="text-align: center"><div>Bitwise OR</div></td>
 </tr>  <tr>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"><div>0</div></td>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"></td>
  <td style="text-align: center"><div>~</div></td>
  <td style="text-align: center"><div>Bitwise_NOT</div></td>
 </tr>  <tr>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"><div>1</div></td>
  <td style="text-align: center"><div>1</div></td>
```

```
    <td style="text-align: center"><div>0</div></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"><div>^</div></td>
    <td style="text-align: center"><div>XOR</div></td>
  </tr>   <tr>
    <td style="text-align: center"> </td>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"><div>&lt;&lt;</div></td>
    <td style="text-align: center"><div>Left Shift</div></td>
  </tr>   <tr>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"></td>
    <td style="text-align: center"><div>&gt;&gt;</div></td>
    <td style="text-align: center"><div>Right Shift</div></td>
  </tr> </tbody></table><ul>
 <li>Consider x=40 and y=80. Binary form of these values are given below.</li>
</ul>
<p>x = 00101000<br>
 y=  01010000</p>
<ul>
 <li>All bit wise operations for x and y are given below.</li>
</ul>
<p>x&amp;y = 00000000 (binary) = 0 (decimal)<br>
 x|y = 01111000 (binary) = 120 (decimal)<br>
 ~x = 11111111111111111111111111 11111111111111111111111111111010111<br>
 .. ..= -41 (decimal)<br>
 x^y = 01111000 (binary) = 120 (decimal)<br>
 x &lt;&lt; 1 = 01010000 (binary) = 80 (decimal)<br>
```

```
 x &gt;&gt; 1 = 00010100 (binary) = 20 (decimal)</p>

<p><strong>Note:</strong></p><ul>

 <li><strong>Bit wise NOT :</strong> Value of 40 in binary
is 00000000000000000000000000000000000000000000000000000101000000000000. So, all 0's are converted into 1's in bit wise NOT operation.</li>

 <li><strong>Bit wise left shift and right shift :</strong> In left shift operation &ldquo;x &lt;&lt; 1 &ldquo;, 1 means that the bits will be left shifted by one place. If we use it as &ldquo;x &lt;&lt; 2 &ldquo;,  then, it means that the bits will be left shifted by 2 places.</li>

</ul>

<h4><strong>Example program for bit wise operators in C:</strong></h4>

<ul>

 <li>In this example program, bit wise operations are performed as shown above and output is displayed in decimal format.</li>

</ul>

<div class="program">#include< stdio.h>

int main()

{

        int m = 40,n = 80,AND_opr,OR_opr,XOR_opr,NOT_opr ;

        AND_opr = (m&n);

        OR_opr = (m|n);

        NOT_opr = (~m);

        XOR_opr = (m^n);

        printf("AND_opr value = %d\n",AND_opr );

        printf("OR_opr value = %d\n",OR_opr );

        printf("NOT_opr value = %d\n",NOT_opr );

        printf("XOR_opr value = %d\n",XOR_opr );

        printf("left_shift value = %d\n", m << 1);

        printf("right_shift value = %d\n", m >> 1);

}</div><h4>Output:</h4>

<div class="output">AND_opr value = 0

OR_opr value = 120

NOT_opr value = -41

XOR_opr value = 120

left_shift value = 80

right_shift value = 20</div>

<h4><strong>6. Conditional or ternary operators in C:</strong></h4>

<div>  <ul>

   <li>Conditional operators return one value if condition is true and returns another value is condition is false.</li>

   <li>This operator is also called as ternary operator.</li>

 </ul></div>
```

Syntax: (Condition? true_value: false_value);

Example : (A > 100 ? 0 : 1);

In above example, if A is greater than 100, 0 is returned. Else 1 is returned. This is equal to if else conditional statements.

<h4>Example program for conditional/ternary operators in C:</h4>

#include< stdio.h>

int main()

{

int x=1, y ;

y = (x ==1 ? 2 : 0) ;

printf("x value is %d\n", x);

printf("y value is %d", y);

}

<h4>Output:</h4> x value is 1

y value is 2

<h3>7. Increment/decrement Operators</h3>

Increment operators are used to increase the value of the variable by one and decrement operators are used to decrease the value of the variable by one in C programs.

Syntax: Increment operator: ++var_name; (or) var_name++;

Decrement operator: – -var_name; (or) var_name – -;

Example: Increment operator : ++ i ; i ++ ;

Decrement operator : - – i ; i – - ;

<h4>Example program for increment operators in C:</h4>

In this program, value of “i” is incremented one by one from 1 up to 9 using “i++” operator and output is displayed as “1 2 3 4 5 6 7 8 9”.

#inlcude< stdio.h>

int main()

{

int i-1;

while(i<10)

{

```
                printf("%d ",i);
                i++;
        }
}</div>
<h4>Output:</h4>
<div class="output">1 2 3 4 5 6 7 8 9</div>
<h4><strong>Example program <strong>for <strong>decrement </strong>operators in C</strong>:</strong></h4>
<ul>
 <li>In this program, value of &amp;ldquo;I&amp;rdquo; is decremented one by one from 20 up to 11 using &amp;ldquo;i–&amp;rdquo; operator and output is displayed as &amp;ldquo;20 19 18 17 16 15 14 13 12 11&amp;rdquo;.</li>
</ul><div class="program">#include< stdio.h>
int main()
{
        int i-20;
        while(i>10)
        {
                printf("%d ",i);
                i--;
        }
}</div>
<h4>Output:</h4>
<div class="output">20 19 18 17 16 15 14 13 12 11</div>
<h4><strong>Difference between pre/post increment &amp;amp; decrement operators in C:</strong></h4>
 <ul>
  <li>Below table will explain the difference between pre/post increment and decrement operators in C.</li>
 </ul>
<table width="750" border="1" cellpadding="2"> <tbody>  <tr>
    <td width="40" style="text-align: center"><div><span data-mce-mark="1"><strong>S.no</strong></span></div></td>
    <td width="132" style="text-align: center"><div><span data-mce-mark="1"><strong>Operator type</strong></span></div></td>
    <td width="75" style="text-align: center"><div><span data-mce-mark="1"><strong>Operator</strong></span></div></td>
    <td width="467" style="text-align: center"><div><span data-mce-mark="1"><strong>Description</strong></span></div></td>
  </tr>  <tr>
    <td style="text-align: center"><div>1</div></td>
    <td style="text-align: center">Pre increment</td>
    <td style="text-align: center">++i<span data-mce-mark="1"><br>
    </span></td>
    <td>Value of i is incremented before assigning it to variable i.</td>
```

```
</tr>  <tr>
 <td style="text-align: center"><div>2</div></td>
 <td style="text-align: center">Post-increment</td>
 <td style="text-align: center">i++<span data-mce-mark="1"><br>
 </span></td>
 <td>Value of i is incremented after assigning it to variable i.</td>
</tr>  <tr>
 <td style="text-align: center"><div>3</div></td>
 <td style="text-align: center">Pre decrement</td>
 <td style="text-align: center">– –i<span data-mce-mark="1"><br>
 </span></td>
 <td>Value of i is decremented before assigning it to variable i.</td>
</tr>  <tr>
 <td style="text-align: center"><div>4</div></td>
 <td style="text-align: center">Post_decrement</td>
 <td style="text-align: center">i– –<span data-mce-mark="1"><br>
 </span></td>
 <td>Value of i is decremented after assigning it to variable i.</td>
</tr> </tbody></table><h3><strong>8. Special Operators in C:</strong></h3>
<div> <ul>
 <li>Below are some of special operators that C language offers.</li>
 </ul></div>
<table width="700" border="1" cellpadding="2"> <tbody>  <tr>
 <td style="text-align: center"><div><strong><span data-mce-mark="1"> S.no</span></strong></div></td>
 <td style="text-align: center"><div><strong><span data-mce-mark="1">Operators</span></strong></div></td>
 <td style="text-align: center"><div><strong><span data-mce-mark="1">Description</span></strong></div></td>
</tr>  <tr>
 <td style="text-align: center"><div>1</div></td>
 <td style="text-align: center"><div>&amp;amp;</div></td>
 <td><div>This is used to get the address of the variable.</div>
  <div>Example : &amp;amp;a will give address of a.</div></td>
</tr>  <tr>
 <td style="text-align: center"><div>2</div></td>
 <td style="text-align: center"><div>*</div></td>
 <td><div>This is used as pointer to a variable.</div>
  <div>Example : * a  where, * is pointer to the variable a.</div></td>
```

```
  </tr>   <tr>
   <td style="text-align: center"><div>3</div></td>
   <td style="text-align: center"><div>Sizeof ()</div></td>
   <td><div>This gives the size of the variable.</div>
    <div>Example : size of (char) will give us 1.</div></td>
  </tr>  </tbody></table>
<h4><strong>Example program for &amp;amp; and * operators in C:</strong></h4><ul>
 <li>In this program, &amp;ldquo;&amp;amp;&amp;rdquo; symbol is used to get the address of the variable and &amp;ldquo;*&amp;rdquo; symbol is used to get the value of the variable that the pointer is pointing to. Please refer <a title="Pointer" href="pointers.html"><strong>Pointer</strong></a> topic to know more about pointers.</li>
</ul><div class="program">#include< stdio.h>
int main()
{
        int *ptr, q;
        q = 50;
        /* address of q is assigned to ptr */
        ptr = &q;
        /* display q's value using ptr variable */
        printf("%d", *ptr);
        return 0;
}</div>
<h4>Output:</h4>
<div class="output">50</div>
<h4><strong>Example program for sizeof() operator in C:</strong></h4>
<ul>
 <li>sizeof() operator is used to find the memory space allocated for each C data types.</li>
</ul>
<div class="program">#include< stdio.h>
int main()
{
        int a;
        char b;
        float c;
        double d;
        printf("Storage size for int data type:%d \n",sizeof(a));
        printf("Storage size for char data type:%d \n",sizeof(b));
        printf("Storage size for float data type:%d \n",sizeof(c));
```

```
        printf("Storage size for double data type:%d\n",sizeof(d));
        return 0;
}</div><h4>Output:</h4>
<div class="output">Storage size for int data type:4
Storage size for char data type:1
Storage size for float data type:4
Storage size for double data type:8</div>
</div>  <div id="Footer">          <ul>
    <li><a href="Variables.html"><<</a></li>
    <li><a href="PrintfAndScanfFunctions.html">1</a></li>
    <li><a href="DataTypes.html">2</a></li>
    <li><a href="Constants.html">3</a></li>
    <li><a href="Variables.html">4</a></li>
    <li><a href="OperatorsAndExpressions.html">5</a></li>
    <li><a href="DecisionControlStatements.html">6</a></li>
    <li><a href="LoopControlStatements.html">7</a></li>
    <li><a href="CaseControlStatements.html">8</a></li>
    <li><a href="Arrays.html">9</a></li>
    <li><a href="String.html">10</a></li>
    <li><a href="Pointers.html">11</a></li>
    <li><a href="Functions.html">12</a></li>
    <li><a href="Structures.html">13</a></li>
    <li><a href="DecisionControlStatements.html">>></a></li>
  </ul>  </div></div></body></html>
```

DecisionControlStatements.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Decision Control Statements</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content"><h1 style="text-align:center;">Decision Control Statements</h1>

   <ul>

    <li>In decision control statements (C if else and nested if), group of statements are executed when condition is true.  If condition is
false, then else part statements are executed.</li>
```

```
 <li>There are 3 types of decision making control statements in C language. They are,
  <ol>
   <li>  if statements</li>
   <li>  if else statements</li>
   <li>  nested if statements</li>
  </ol>
 </li>
</ul>
<h4><strong>&amp;ldquo;If&amp;rdquo;, &amp;ldquo;else&amp;rdquo; and &amp;ldquo;nested if&amp;rdquo; decision control statements in C:</strong></h4>
<ul>
 <li>Syntax for each C decision control statements are given in below table with description.</li>
</ul>
<table width="900" border="1" align="center" cellpadding="3">
 <tbody>
  <tr>
   <td width="149" style="text-align: center"><strong>Decision control statements</strong></td>
   <td width="210" style="text-align: center"><strong>Syntax</strong></td>
   <td width="507" style="text-align: center"><strong>Description</strong></td>
  </tr>
  <tr>
   <td style="text-align: center"><strong>if</strong></td>
   <td style="text-align: justify">if (condition){<br>Statements;<br> }</td>
   <td style="text-align: justify">In these type of statements, if condition is true, then respective block of code is executed.</td>
  </tr>
  <tr>
   <td style="text-align: center"><strong>if…else</strong></td>
   <td style="text-align: justify"><p>if(condition){ <br>Statement1; <br>Statement2;<br>} <br> else{<br>Statement3;<br>
Statement4;<br> }</p></td>
   <td style="text-align: justify">In these type of statements, group of statements are executed when condition is true.  If condition is
false, then else part statements are executed.</td>
  </tr>
  <tr>
   <td height="125" style="text-align: center"><strong><strong>nested if</strong></strong></td>
   <td style="text-align: justify">if (condition1){ <br>Statement1;<br> }<br>
    else_if(condition2){<br>
    Statement2;<br>} <br>
   else<br>Statement 3;</td>
```

<td style="text-align: justify">If condition 1 is false, then condition 2 is checked and statements are executed if it is true. If condition 2 also gets failure, then else part is executed.</td>

</tr>

</tbody>

</table>

<h4>Example program for if statement in C:</h4>

 In “if” control statement, respective block of code is executed when condition is true.

int main()

{

int m-40,n-40;

if(m--n)

{

printf("m and n are equal.");

}

}

<h4>Output:</h4>

m and n are equal.

<h4>Example program for if else statement in C:</h4>

 In C if else control statement, group of statements are executed when condition is true. If condition is false, then else part statements are executed.

#include<stdio.h&rt;

int main()

{

int m-40,n-20;

if(m--n)

{

printf("m and n are equal");

}

else

{

printf("m and n are not equal");

}

}

<h4>Output:</h4>

m and n are not equal

<h4>Example program for nested if statement in C: </h4>

In “nested if” control statement, if condition 1 is false, then condition 2 is checked and statements are executed if it is true.

If condition 2 also gets failure, then else part is executed.

#include<stdio.h>

int main()

{

int m-40,n-20;

if(m>n)

printf("m is greater than n");

else if(m<n)

printf("m is less than n");

else

printf("m is equal to n");

}

<h4>Output:</h4>

m is greater than n

<<

1

2

3

4

5

6

7

8

9

10

11

12

13

>>

 </body></html>

LoopControlStatements.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Loop Control Statements</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content"><h1 style="text-align:center;">Loop Control Statements</h1>

   <p> Loop control statements in C are used to perform looping operations until the given condition is true. Control comes out of the loop
statements once condition becomes false.</p>

   <h4><strong>Types of loop control statements in C:</strong></h4>
```

```
<p>There are 3 types of loop control statements in C language. They are,</p>
<ol start="1">
 <li>for</li>
 <li>while</li>
 <li>do-while</li>
</ol>
<ul>
 <li>Syntax for each C loop control statements are given in below table with description.</li>
</ul>
<table width="700" border="1" cellpadding="5">
 <tbody>
  <tr>
   <td width="50" colspan="1" rowspan="1" style="text-align: center"><strong>S.no</span></strong></td>
   <td width="80" colspan="1" rowspan="1" style="text-align: center"><div><strong>Loop Name</span></strong></div></td>
   <td width="195" colspan="1" rowspan="1" style="text-align: center"><div><strong>Syntax</span></strong></div></td>
   <td width="331" colspan="1" rowspan="1" style="text-align: center"><div><strong>Description</span></strong></div></td>
  </tr>
  <tr>
   <td style="text-align: center">1</td>
   <td style="text-align: center">for</td>
   <td style="text-align: left" class="syntax">
   for (exp1; exp2; expr3)
{
  statements;
}
</td>
   <td style="text-align: left">Where,<br>
    exp1 – variable initialization<br>
    ( Example: i=0, j=2, k=3 )<br>
    exp2 – condition checking<br>
    ( Example: i&gt;5, j&lt;3, k=3 )<br>
    exp3 – increment/decrement<br>
    ( Example: ++i, j–, ++k )</td>
  </tr>  <tr>
   <td style="text-align: center">2</td>
   <td style="text-align: center"><p>while</p></td>
```

<td style="text-align: left" class="syntax">while (condition)

{

statements;

}</td>

<td style="text-align: left">where,

condition might be a>5, i<10</td>

</tr> <tr>

<td style="text-align: center">3</td>

<td style="text-align: center"> do while </td>

<td style="text-align: left" class="syntax">do

{

statements;

}

while(condition);</td>

<td style="text-align: left">where,

condition might be a>5, i<10</td>

</tr> </tbody> </table>

<h4>Example program (for loop) in C:</h4>

In for loop control statement, loop is executed until condition becomes false

int main()

{

int i;

for(i-0;i<10;i++)

{

printf("%d ",i);

}

} <h4>Output:</h4>

0 1 2 3 4 5 6 7 8 9

<h4>Example program (while loop) in C:</h4>

In while loop control statement, loop is executed until condition becomes false.

int main()

{

int i-3;

while(i<10)

{

printf("%d\n",i);

```
    i++;
  }
}</div><h4>Output:</h4><div class="output">3 4 5 6 7 8 9</div>
<h4><strong>Example program (do while loop) in C:</strong></h4>
<p>    In do..while loop control statement, while loop is executed irrespective of the condition for first time. Then 2nd time onwards, loop is executed until condition becomes false.</p>
 <div class="program">int main()
{
  int i-1;
  do{
    printf("value of i is %d\n",i);
    i++;
  }while(i&lt;-4 &amp;amp;&amp;amp; i&gt;-2);
}</div>
<h4>Output:</h4><div class="output">Value of i is 1
Value of i is 2
Value of i is 3
Value of i is 4</div>
<h4><strong>Difference between while &amp;amp; do while loops in C:</strong></h4>
<table width="600" border="1" cellpadding="2"> <tbody>  <tr>
   <td width="38" colspan="1" rowspan="1" style="text-align: center"><span data-mce-mark="1"><strong>S.no</strong></span></td>
   <td width="267" colspan="1" rowspan="1" style="text-align: center"><div><span data-mce-mark="1"><strong>while</strong></span></div></td>
   <td width="267" colspan="1" rowspan="1" style="text-align: center"><div><span data-mce-mark="1"><strong>do while</strong></span></div></td>
  </tr>  <tr>
   <td style="text-align: center">1</td>
   <td style="text-align: justify">Loop is executed only when condition is true.</td>
   <td style="text-align: justify">Loop is executed for first time irrespective of the condition. After <span style="text-align: justify">executing while loop for first time, then condition is checked.</span></td>
  </tr> </tbody></table> </div> <div id="Footer">  <ul>
    <li><a href="DecisionControlStatements.html">&lt;</a></li>
    <li><a href="PrintfAndScanfFunctions.html">1</a></li>
    <li><a href="DataTypes.html">2</a></li>
    <li><a href="Constants.html">3</a></li>
    <li><a href="Variables.html">4</a></li>
    <li><a href="OperatorsAndExpressions.html">5</a></li>
```

```
      <li><a href="DecisionControlStatements.html">6</a></li>
      <li><a href="LoopControlStatements.html">7</a></li>
      <li><a href="CaseControlStatements.html">8</a></li>
      <li><a href="Arrays.html">9</a></li>
      <li><a href="String.html">10</a></li>
      <li><a href="Pointers.html">11</a></li>
      <li><a href="Functions.html">12</a></li>
      <li><a href="Structures.html">13</a></li>
      <li><a href="CaseControlStatements.html">&gt;</a></li>
    </ul>
  </div>
</div>
</body>
</html>
```

CaseControlStatements.html

```
<!doctype html><html><head><meta charset="utf-8">

<title>Case Control Statements</title>

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

</head><body><div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul></div>

 <div id="Content"><h1 style="text-align:center;">Case Control Statements</h1>

   <p>The statements which are used to execute only specific block of statements in a series of blocks are called case control
statements.</p>

   <p>There are 4 types of case control statements in C language. They are,</p>

   <ol>

    <li>switch</li>

    <li>break</li>

    <li>continue</li>

    <li>goto</li>

   </ol>   <h4><strong>1. switch case statement in C:</strong></h4>
```

```
<ul>  <li>Switch case statements are used to execute only specific case statements based on the switch expression.</li>
 <li>Below is the syntax for switch case statement.</li>  </ul>
<div class="syntax">switch(expression)
{
  case label1: statements;
  break;
  case label2: statements;
  break;
  default: statements;
}</div>
  <h4><strong>Example program for switch..case statement in C:</strong></h4>
  <div class="program">int main()
{
  int value = 3;
  switch(value)
  {
    case1: printf("Value is 1 \n");
    break;
    case2: printf("Value is 2 \n");
    break;
    case3: printf("Value is 3 \n");
    break;
    case4: printf("Value is 4 \n");
    break;
    default: printf("Value is other than 1,2,3,4 \n");
  }
  return 0;
}</div>  <h4><strong>Output:</strong></h4>
  <div class="output">Value is 3 </div>
  <h4><strong>2. break statement in C:</strong></h4>
  <ul>
   <li>Break statement is used to terminate the while loops, switch case loops and for loops from the subsequent execution.</li>
   <li class="syntax">Syntax: break;</li>
  </ul>
  <h4><strong>Example program for break statement in C:</strong></h4>
  <div class="program">int main()
```

```
{
   int i;
   for(i=0;i&lt;10;i++)
   {
      if(i==5)
      {
         printf("\nComing out of for loop when i = 5");
         break;
      }
      printf("%d ",i);
   }
}</div><h4>Output:</h4>
<div class="output">0 1 2 3 4
Coming out of for loop when i = 5</div>
<h4><strong>3. Continue statement in C:</strong></h4>
<ul>  <li>Continue statement is used to continue the next iteration of for loop, while loop and do-while loops.  So, the remaining statements are skipped within the loop for that particular iteration.</li>
  <li class="syntax">Syntax : continue;</li></ul><h4><strong>Example program for continue statement in C:</strong></h4>
<div class="program">int main()
{
   int i;
   for(i=0;i&lt;10;i++)
   {
      if(i==5 || i==6)
      {
         printf("\nSkipping %d from display usng continue statement \n",i);
         continue;
      }
      printf("%d ",i);
   }
}</div><h4>Output:</h4>
<div class="output">0 1 2 3 4
Skipping 5 from display using continue statement
Skipping 6 from display using continue statement
7 8 9</div>
<h4><strong>4. goto statement in C:</strong></h4>
<ul>  <li>goto statements is used to transfer the normal flow of a program to the specified label in the program.</li>
```

```
 <li>Below is the syntax for goto statement in C.</li></ul>
<div class="syntax">{
   .....
   go to label;
   .....
   .....
   LABEL;
   statements;
}</div><h4><strong>Example program for goto statement in C:</strong></h4>
<div class="program">int main()
{
   int i;
   for(i=0;i&lt;10;i++)
   {
      if(i==5)
      {
         printf("\nWe are using goto statement when i = 5");
         goto HAI;
      }
      prinf("%d ",i);
   }
   HAI : printf("\nNow, we are inside label name\"hai\" \n");
}</div><h4>Output:</h4>
<div class="output">0 1 2 3 4
We are using goto statement when i = 5
Now, we are inside label name "hai"</div>  </div>
 <div id="Footer">     <ul>
      <li><a href="LoopControlStatements.html">&lt;</a></li>
      <li><a href="PrintfAndScanfFunctions.html">1</a></li>
      <li><a href="DataTypes.html">2</a></li>
      <li><a href="Constants.html">3</a></li>
      <li><a href="Variables.html">4</a></li>
      <li><a href="OperatorsAndExpressions.html">5</a></li>
      <li><a href="DecisionControlStatements.html">6</a></li>
      <li><a href="LoopControlStatements.html">7</a></li>
      <li><a href="CaseControlStatements.html">8</a></li>
```

```
      <li><a href="Arrays.html">9</a></li>
      <li><a href="String.html">10</a></li>
      <li><a href="Pointers.html">11</a></li>
      <li><a href="Functions.html">12</a></li>
      <li><a href="Structures.html">13</a></li>
      <li><a href="Arrays.html">&gt;</a></li>
   </ul>
  </div>
</div>
</body>
</html>
```

Arrays.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Arrays</title>

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

</head>

<body>

<div id="Container">

  <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

  <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

    <ul>

      <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

       <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

   </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

       <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

       <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

       <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content"><h1 style="text-align:center;">Arrays</h1>

   <p>C Array is a collection of variables belongings to the same data type. You can store group of data of same data type in an array.</p>

   <ul>
```

```
 <li>Array might be belonging to any of the data types</li>
 <li>Array size must be a constant value.</li>
 <li>Always, Contiguous (adjacent) memory locations are used to store array elements in memory.</li>
 <li>It is a best practice to initialize an array to zero or null while declaring, if we don&rsquo;t assign any values to array.</li>
</ul>
<h4><strong>Example for C Arrays:</strong></h4>
<ul>
 <li>int a[10];      // integer array</li>
 <li>char b[10];   // character array   i.e. string</li>
</ul>
<h4><strong>Types of C arrays:</strong></h4>
<p>There are 2 types of C arrays. They are,</p>
<ol>
 <li>One dimensional array</li>
 <li>Multi dimensional array
  <ol>
   <li>Two dimensional array</li>
   <li>Three dimensional array, four dimensional array etc…</li>
  </ol>    </li>   </ol>
<h4><strong>1. O</strong><strong>ne dimensional array in C:</strong></h4>
<ul>
 <li class="syntax">Syntax : data-type arr_name[array_size];</li>
</ul>
<table width="900" border="1" cellpadding="2">
 <tbody>      <tr>
   <td style="text-align: center"><div><strong>Array declaration</strong></div></td>
   <td style="text-align: center"><div><strong>Array initialization</strong></div></td>
   <td style="text-align: center"><div><strong>Accessing array</strong></div></td>
  </tr>      <tr>
   <td>Syntax: data_type arr_name[arr_size];</td>
   <td>data_type arr_name [arr_size]=
(value1, value2,….);</td>
   <td>arr_name[index];</td>
  </tr>      <tr>
   <td>int age [5];</td>
   <td>int age[5]={0, 1, 2, 3, 4};</td>
```

<td>age[0];_/*0_is_accessed*/age[1];_/*1_is_accessed*/age[2];_/*2_is_accessed*/</td>

</tr> <tr>

<td>char str[10];</td>

<td>char str[10]={‘H’,‘a’,‘i’}; (or)char str[0] = ‘H’;char str[1] = ‘a’;

char str[2] = ‘i; </td>

<td>str[0];_/*H is accessed*/str[1]; /*a is accessed*/str[2]; /* i is accessed*/</td>

</tr>

</tbody>

</table>

<h4>Example program for one dimensional array in C:</h4>

int main()

{

int i;

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

//declaring and initializing an array in C

//To initialize all array elements to 0, use int arr[5]={0};

/* Above array can be initialized as below also

arr[0] = 10;

arr[1] = 20;

arr[2] = 30;

arr[3] = 40;

arr[4] = 50; */

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

{

//Accessing each variable

prinf("Value of arr[%d] is %d \n", i, arr[i]);

}

} <h4>Output:</h4>

Value of arr[0] is 10

Value of arr[1] is 20

Value of arr[2] is 30

Value of arr[3] is 40

Value of arr[4] is 50

<h4>2. Two dimensional array in C:</h4>

Two dimensional array is nothing but array of array.

```
 <li>syntax : data_type array_name[num_of_rows][num_of_column]</li>
</ul>
<table width="900" border="1" cellpadding="2">
 <tbody>
  <tr>
   <td style="text-align: center"><strong>S.no</strong></td>
   <td style="text-align: center"><div><strong>Array declaration</strong></div></td>
   <td style="text-align: center"><div><strong>Array initialization</strong></div></td>
   <td style="text-align: center"><div><strong>Accessing array</strong></div></td>
  </tr>
  <tr>
   <td style="text-align: center">1</td>
   <td>Syntax: data_type arr_name [num_of_rows][num_of_column];</td>
   <td>data_type arr_name[2][2] = {{0,0},{0,1},{1,0},{1,1}};</td>
   <td>arr_name[index];</td>
  </tr>
  <tr>
   <td style="text-align: center">2</td>
   <td>Example:int arr[2][2];</td>
   <td>int arr[2][2] = {1,2, 3, 4};</td>
   <td>arr [0] [0] = 1; arr [0] ]1] = 2;arr [1][0]  = 3;
    <p>arr [1] [1] = 4;</p></td>
  </tr>
 </tbody>
</table>
<h4><strong>Example program for two dimensional array in C:</strong></h4>
<div class="program">int main()
{
   int i,j;
   //declaring and initializing array
   int arr[2][2] = {10,20,30,40};
   /*Above array can be initialized as below also
   arr[0][0] = 10;
   arr[0][1] = 20;
   arr[1][0] = 30;
   arr[1][1] = 40; */
```

```
  for(i=0;i&lt;2;i++)
  {
     for(j=0;j&lt;2;j++)
     {
        //Accessing variables
        printf("value of arr[%d][%d] : %d\n",i,j,arr[i][j]);
     }
  }
}</div><h4>Output:</h4>
<div class="output">value of arr[0][0] is 10
value of arr[0][1] is 20
value of arr[1][0] is 30
value of arr[1][1] is 40</div>
 </div>
 <div id="Footer">
    <ul>
      <li><a href="CaseControlStatements.html"><</a></li>
      <li><a href="PrintfAndScanfFunctions.html">1</a></li>
      <li><a href="DataTypes.html">2</a></li>
      <li><a href="Constants.html">3</a></li>
      <li><a href="Variables.html">4</a></li>
      <li><a href="OperatorsAndExpressions.html">5</a></li>
      <li><a href="DecisionControlStatements.html">6</a></li>
      <li><a href="LoopControlStatements.html">7</a></li>
      <li><a href="CaseControlStatements.html">8</a></li>
      <li><a href="Arrays.html">9</a></li>
      <li><a href="String.html">10</a></li>
      <li><a href="Pointers.html">11</a></li>
      <li><a href="Functions.html">12</a></li>
      <li><a href="Structures.html">13</a></li>
      <li><a href="String.html">></a></li>
   </ul>
 </div>
</div>
</body>
</html>
```

String.html

```
<!doctype html>

<html><head><meta charset="utf-8">

<title>String</title>

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

</head><body><div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu"><ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>     <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>   </ul>  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

   <ul>      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

     <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>     </ul>  </li>

  <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li></ul></div>

 <div id="Content"><h1 style="text-align:center;">String</h1>   <ul>

    <li>C Strings are nothing but array of characters ended with null character (&lsquo;\0&rsquo;).</li>

    <li>This null character indicates the end of the string.</li>

    <li>Strings are always enclosed by double quotes. Whereas, character is enclosed by single quotes in C.</li>

  </ul>   <h4><strong>Example for C string:</strong></h4>

  <ul>     <li>char string[20] = {'L','a','n','g','u','a','g','e','T','u','t','o','r', &lsquo;\0&rsquo;}; (or)</li>

    <li>char string[20] = &ldquo;LanguageTutor&rdquo;; (or)</li>

    <li>char string [] = &ldquo;LanguageTutor&rdquo;;</li>

    <li>Difference between above declarations are, when we declare char as &ldquo;string[20]&ldquo;, 20 bytes of memory space is
allocated for holding the string value.</li>

    <li>When we declare char as &ldquo;string[]&rdquo;, memory space will be allocated as per the requirement during execution of the
program.</li>

  </ul>   <h4><strong>Example program for C string:</strong></h4><div class="program">#include&lt;stdio.h&gt;

int main()

{

  char string[20] - "sgmlanguagetutor.edu";

  printf("The string is: %s \n",string);
```

```
   return 0;
}</div><h4>Output:</h4>
<div class="output">The string is: sgmlanguagetutor.edu</div>
<h4><strong>C String functions:</strong></h4>
<ul>  <li>String.h header file supports all the string functions in C language. All the string functions are given below.</li>
</ul><table width="1000" border="1" align="center" cellpadding="2">  <tbody>   <tr>
    <td colspan="1" rowspan="1" style="text-align: center"><strong>S.no</strong></td>
    <td colspan="1" rowspan="1" style="text-align: center"><div><strong>String functions</strong></div></td>
    <td colspan="1" rowspan="1" style="text-align: center"><div><strong>Description</strong></div></td>   </tr>   <tr>
    <td style="text-align: center">1</td>
    <td style="text-align: center"><strong>strcat ( )</strong></td>
    <td> Concatenates str2 at the end of str1.</td>   </tr>   <tr>
    <td style="text-align: center">2</td>
    <td style="text-align: center"><strong>strncat ( )</strong></td>
    <td> appends a portion of string to another</td>   </tr>   <tr>
    <td style="text-align: center">3</td>
    <td style="text-align: center"><strong>strcpy ( )</strong></td>
    <td> Copies str2 into str1</td>
  </tr>   <tr>
    <td style="text-align: center">4</td>
    <td style="text-align: center"><strong>strncpy ( )</strong></td>
    <td> copies given number of characters of one string to another</td>
  </tr>   <tr>
    <td style="text-align: center">5</td>
    <td style="text-align: center"><strong>strlen ( )</strong></td>
    <td> gives the length of str1.</td>
  </tr>   <tr>
    <td style="text-align: center">6</td>
    <td style="text-align: center"><strong>strcmp ( )</strong></td>
    <td> Returns 0 if str1 is same as str2. Returns &amp;lt;0 if strl &amp;lt; str2. Returns &amp;gt;0 if str1 &amp;gt; str2.</td>
  </tr>   <tr>
    <td style="text-align: center">7</td>
    <td style="text-align: center"><strong>strcmpi_( )</strong></td>
    <td> Same as strcmp() function. But, this function negotiates case.  &amp;ldquo;A&amp;rdquo; and &amp;ldquo;a&amp;rdquo; are treated as same.</td>
  </tr>   <tr>
    <td style="text-align: center">8</td>
```

```
 <td style="text-align: center"><strong>strchr ( )</strong></td>
 <td> Returns pointer to first occurrence of char in str1.</td>
</tr>  <tr>
 <td style="text-align: center">9</td>
 <td style="text-align: center"><strong>strrchr ( )</strong></td>
 <td> last occurrence of given character in a string is found</td>
</tr>  <tr>
 <td style="text-align: center">10</td>
 <td style="text-align: center"><strong>strstr ( )</strong></td>
 <td> Returns pointer to first occurrence of str2 in str1.</td>
</tr>  <tr>
 <td style="text-align: center">11</td>
 <td style="text-align: center"><strong>strrstr ( )</strong></td>
 <td> Returns pointer to last occurrence of str2 in str1.</td>
</tr>  <tr>
 <td style="text-align: center">12</td>
 <td style="text-align: center"><strong>strdup ( )</strong></td>
 <td> duplicates the string</td>
</tr>  <tr>
 <td style="text-align: center">13</td>
 <td style="text-align: center"><strong>strlwr ( )</strong></td>
 <td> converts string to lowercase</td>
</tr>  <tr>
 <td style="text-align: center">14</td>
 <td style="text-align: center"><strong>strupr ( )</strong></td>
 <td> converts string to uppercase</td>
</tr>  <tr>
 <td style="text-align: center">15</td>
 <td style="text-align: center"><strong>strrev ( )</strong></td>
 <td> reverses the given string</td>
</tr>  <tr>
 <td style="text-align: center">16</td>
 <td style="text-align: center"><strong>strset ( )</strong></td>
 <td> sets all character in a string to given character</td>
</tr>  <tr>
 <td style="text-align: center">17</td>
```

```
    <td style="text-align: center"><strong>strnset ( )</strong></td>
    <td> It sets the portion of characters in a string to given character</td>
  </tr>   <tr>
    <td style="text-align: center">18</td>
    <td style="text-align: center"><strong>strtok ( )</strong></td>
    <td> tokenizing given string using delimiter</td>
  </tr> </tbody></table> </div>
 <div id="Footer">    <ul>      <li><a href="Arrays.html">&lt;</a></li>
      <li><a href="PrintfAndScanfFunctions.html">1</a></li>
      <li><a href="DataTypes.html">2</a></li>
      <li><a href="Constants.html">3</a></li>
      <li><a href="Variables.html">4</a></li>
      <li><a href="OperatorsAndExpressions.html">5</a></li>
      <li><a href="DecisionControlStatements.html">6</a></li>
      <li><a href="LoopControlStatements.html">7</a></li>
      <li><a href="CaseControlStatements.html">8</a></li>
      <li><a href="Arrays.html">9</a></li>
      <li><a href="String.html">10</a></li>
      <li><a href="Pointers.html">11</a></li>
      <li><a href="Functions.html">12</a></li>
      <li><a href="Structures.html">13</a></li>
      <li><a href="Pointers.html">&gt;</a></li>
   </ul>
</div>
</div>
</body>
</html>
```

Pointers.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Pointers</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content"><h1 style="text-align:center;">Pointers</h1>

   <ul>

    <li>C Pointer is a variable that stores/points the address of another variable. C Pointer is used to allocate memory dynamically i.e. at
run time. The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.</li>
```

```
</ul>
<p class="syntax">Syntax:  data_type *var_name;
Example : int *p;
          char *p;</p>
<ul>
 <li>Where, * is used to denote that &amp;ldquo;p&amp;rdquo; is pointer variable and not a normal variable.</li>
</ul>
<h4><strong>Key points to remember about pointers in C:</strong></h4>
<ul>
 <li>Normal variable stores the value whereas pointer variable stores the address of the variable.</li>
 <li>The content of the C pointer always be a whole number i.e. address.</li>
 <li>Always C pointer is initialized to null, i.e. int *p = null.</li>
 <li>The value of null pointer is 0.</li>
</ul>
<ul>
 <li>&amp;amp; symbol is used to get the address of the variable.</li>
 <li>* symbol is used to get the value of the variable that the pointer is pointing to.</li>
 <li>If pointer is assigned to NULL, it means it is pointing to nothing.</li>
 <li>Two pointers can be subtracted to know how many elements are available between these two pointers.</li>
</ul>
<ul>
 <li>But, Pointer addition, multiplication, division are not allowed.</li>
 <li>The size of any pointer is 2 byte (for 16 bit compiler).</li>
</ul>
<h4><strong>Example program for pointer in C:</strong></h4>
<div class="program">#include&lt;stdio.h&gt;
int main()
{
 int *ptr,q;
 q = 50;
 /* Address of q is assigned to ptr */
 ptr = &amp;amp;q;
 /* display q's value using ptr variable */
 printf("%d",*ptr);
 return 0;
}</div><h4>Output:</h4>
```

```
<div class="output">50</div>
 </div>
 <div id="Footer">
            <ul>
     <li><a href="String.html"><<</a></li>
     <li><a href="PrintfAndScanfFunctions.html">1</a></li>
     <li><a href="DataTypes.html">2</a></li>
     <li><a href="Constants.html">3</a></li>
     <li><a href="Variables.html">4</a></li>
     <li><a href="OperatorsAndExpressions.html">5</a></li>
     <li><a href="DecisionControlStatements.html">6</a></li>
     <li><a href="LoopControlStatements.html">7</a></li>
     <li><a href="CaseControlStatements.html">8</a></li>
     <li><a href="Arrays.html">9</a></li>
     <li><a href="String.html">10</a></li>
     <li><a href="Pointers.html">11</a></li>
     <li><a href="Functions.html">12</a></li>
     <li><a href="Structures.html">13</a></li>
     <li><a href="Functions.html">>></a></li>
   </ul>
 </div>
</div>
</body>
</html>
```

Functions.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Functions</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content"><h1 style="text-align:center;">Functions</h1>

   <div>

    <p>C functions are basic building blocks in a program. All C programs are written using functions to improve re-usability,
understandability and to keep track on them. </p>
```

```
<h4><strong>1. What is C function?</strong></h4>

<p>  A large C program is divided into basic building blocks called C function. C function contains set of instructions enclosed by &amp;ldquo;{ }&amp;rdquo; which performs specific operation in a C program. Actually, Collection of these functions creates a C program.</p>

<h4><strong>2. Uses of C functions:</strong></h4>

<ul>

 <li>C functions are used to avoid rewriting same logic/code again and again in a program.</li>

 <li>There is no limit in calling C functions to make use of same functionality wherever required.</li>

 <li>We can call functions any number of times in a program and from any place in a program.</li>

 <li>A large C program can easily be tracked when it is divided into functions.</li>

 <li>The core concept of C functions are, re-usability, dividing a big task into small pieces to achieve the functionality and to improve understandability of very large C programs.</li>

</ul>

<h4><strong>3. C function declaration, function call and function definition:</strong></h4>

<p>There are 3 aspects in each C function. They are,</p>

<ul>

 <li>Function declaration or prototype  - This informs compiler about the function name, function parameters and  return value&amp;rsquo;s data type.</li>

 <li>Function call – This calls the actual function</li>

 <li>Function definition – This contains all the statements to be executed.</li>

</ul>

<div>

 <table width="500" border="1" cellpadding="2">

  <thead>

   <tr>

    <td width="38" style="text-align: center"><strong>S.no</strong></td>

    <td width="122" style="text-align: center"><strong>C function aspects</strong></td>

    <td width="324" style="text-align: center"><strong>syntax</strong></td>

   </tr>

   <tr>

    <td style="text-align: center">1</td>

    <td>function definition</td>

    <td>return_type function_name ( arguments list )<br>

     { Body of function; }</td>

   </tr>

   <tr>

    <td style="text-align: center">2</td>

    <td>function call</td>

    <td>function_name ( arguments list );</td>
```

```
          </tr>
          <tr>
            <td style="text-align: center">3</td>
            <td>function declaration</td>
            <td>return_type function_name ( argument list );</td>
          </tr>
        </thead>
      </table>
    </div>
  </div>
  <h4><strong>Simple example program for C function:</strong></h4>
  <ul>
    <li>As you know, functions should be declared and defined before calling in a C program.</li>
    <li>In the below program, function &amp;ldquo;square&amp;rdquo; is called from main function.</li>
    <li>The value of &amp;ldquo;m&amp;rdquo; is passed as argument to the function &amp;ldquo;square&amp;rdquo;. This value is multiplied by itself in this function and multiplied value &amp;ldquo;p&amp;rdquo; is returned to main function from function &amp;ldquo;square&amp;rdquo;.</li>
  </ul>
  <div class="program">#include&amp;lt;stdio.h&amp;gt;
//function prototype, also called funcion declaration
float square(float x);
//main function, program starts from here
int main()
{
  float m,n;
  printf("\n Enter some number for finding square \n);
  scanf("%f",&amp;amp;m);
  //function call
  n-square(m);
  printf("\n Square of the given number %f is %f",m,n);
}
float square(float x) //function definition
{
  float p;
  p-x*x;
  return(p);
}</div>
<h4>Output:</h4>
```

Enter some number of finding square

2

Square of the given number 2.000000 is 4.000000

<h4> How to call C functions in a program?</h4>

There are two ways that a C function can be called from a program. They are,

Call by value

Call by reference

<h4>1. Call by value:</h4>

In call by value method, the value of the variable is passed to the function as parameter.

The value of the actual parameter can not be modified by formal parameter.

Different Memory is allocated for both actual and formal parameters. Because, value of actual parameter is copied to formal parameter.

Note:

Actual parameter – This is the argument which is used in function call.

Formal parameter – This is the argument which is used in function definition

<h4>Example program for C function (using call by value):</h4>

In this program, the values of the variables “m” and “n” are passed to the function “swap”.

These values are copied to formal parameters “a” and “b” in swap function and used.

#include<stdio.h>

//function prototype, also called function declaration

void swap(int a, int b);

int main()

{

int m-22,n-44;

//calling swap function by value

printf("values before swap m - %d /nand n - %d",m,n);

```
   swap(m,n);
}
void swap(int a, int b)
{
   int tmp;
   tmp - a;
   a - b;
   b - tmp;
   printf("\n values after swap m - %d\n and n - %d",a,b);
}</div>
<h4>Output:</h4>
<div class="output">Values befor swap m = 22
and n = 44
values after swap m = 44
and n = 22</div>
<h4><strong>Call by reference:</strong></h4>
<div>
 <ul>
   <li>In call by reference method, the address of the variable is passed to the function as parameter.</li>
   <li>The value of the actual parameter can be modified by formal parameter.</li>
   <li>Same memory is used for both actual and formal parameters since only address is used by both parameters.</li>
 </ul>
</div>
<h4><strong>Example program for C function (using call by reference):</strong></h4>
<ul>
 <li>In this program, the address of the variables &amp;ldquo;m&amp;rdquo; and &amp;ldquo;n&amp;rdquo; are passed to the function
&amp;ldquo;swap&amp;rdquo;.</li>
 <li>These values are not copied to formal parameters &amp;ldquo;a&amp;rdquo; and &amp;ldquo;b&amp;rdquo; in swap function.</li>
 <li>Because, they are just holding the address of those variables.</li>
 <li>This address is used to access and change the values of the variables.</li>
</ul><div class="program">#include&amp;lt;stdio.h&amp;gt;
// function prototype, also called function declaration
void swap(int *a, int *b);
int main()
{
   int m - 22, n - 44;
   //calling swap function by reference
```

```
   printf("Values before swap m - %d \n and n - %d",m,n);
   swap(&amp;m, &amp;n);
}
void swap(int *a, int *b)
{
   int tmp;
   tmp - *a;
   *a - *b;
   *b - tmp;
   printf("\n values after swap a - %d \nand b - %d", *a,*b);
}</div><h4>Output:</h4>
<div class="output">values before swap m = 22
and n = 44
values after swap a = 44
and b=22</div>
 </div>
 <div id="Footer">
                <ul>
                <li><a href="Pointers.html">&lt;</a></li>
      <li><a href="PrintfAndScanfFunctions.html">1</a></li>
      <li><a href="DataTypes.html">2</a></li>
      <li><a href="Constants.html">3</a></li>
      <li><a href="Variables.html">4</a></li>
      <li><a href="OperatorsAndExpressions.html">5</a></li>
      <li><a href="DecisionControlStatements.html">6</a></li>
      <li><a href="LoopControlStatements.html">7</a></li>
      <li><a href="CaseControlStatements.html">8</a></li>
      <li><a href="Arrays.html">9</a></li>
      <li><a href="String.html">10</a></li>
      <li><a href="Pointers.html">11</a></li>
      <li><a href="Functions.html">12</a></li>
      <li><a href="Structures.html">13</a></li>
      <li><a href="Structures.html">&gt;</a></li>
   </ul>  </div></div></body></html>
```

Structures.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Structures</title>

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

</head>

<body>

<div id="Container">

 <div id="Header"><img src="../../Logo/Logo.png" alt="Logo" width="103" height="103" title="Logo"/><img id="titile"
src="../../Logo/title.png" width="796" height="110" alt="Title"/></div>

 <div id="cssmenu">

<ul>

  <li><a href="../welcome.html"><span>Welcome</span></a></li>

  <li class="active has-sub"><a href="#.html"><span>Account</span></a>

   <ul>

    <li class="has-sub"><a href="#.html"><span>Profile</span></a></li>

     <li class="has-sub"><a href="#.html"><span>Settings</span></a></li>

  </ul>

  </li>

  <li class="active has-sub"><a href="#"><span>Learn C</span></a>

    <ul>

      <li class="has-sub"><a href="../Basics.html"><span>Basics</span></a></li>

      <li class="has-sub"><a href="../Topics.html"><span>Topics</span></a></li>

      <li class="has-sub"><a href="#.html"><span>Questions</span></a></li>

    </ul>

  </li>

   <li class="last"><a href="../Contacts.html"><span>Contact</span></a></li>

  <li class="last"><a href="#"><span>Log Out</span></a></li>

</ul>

</div>

 <div id="Content"><h1 style="text-align:center;">Structures</h1>

   <p> C Structure is a collection of different data types which are grouped together and each element in a C structure is called
member.</p>

   <ul>
```

```
 <li>If you want to access structure members in C, structure variable should be declared.</li>
 <li>Many structure variables can be declared for same structure and memory will be allocated for each separately.</li>
 <li>It is a best practice to initialize a structure to null while declaring, if we don&rsquo;t assign any values to structure members.</li>
</ul>
<h4><strong>Difference between C variable, C array and C structure:</strong></h4>
<ul>
 <li>A normal C variable can hold only one data of one data type at a time.</li>
 <li>An array can hold group of data of same data type.</li>
 <li>A structure can hold group of data of different data types</li>
 <li>Data types can be int, char, float, double and long double etc.</li>
</ul>
<div>
 <table width="900" border="1" align="center" cellpadding="2">
  <tbody>
   <tr>
    <td width="91" rowspan="2" style="text-align: center"><strong>Datatype</strong></td>
    <td colspan="2" style="text-align: center"><h4><strong><strong>C variable</strong></strong></h4></td>
    <td colspan="2" style="text-align: center"><h4><strong>C array</strong></h4></td>
    <td colspan="2" style="text-align: center"><h4><strong>C structure</strong></h4></td>
   </tr>
   <tr>
    <td width="107" style="text-align: center">Syntax</td>
    <td width="103" style="text-align: center">Example</td>
    <td width="119" style="text-align: center">Syntax</td>
    <td width="121" style="text-align: center">Example</td>
    <td width="165" style="text-align: center">Syntax</td>
    <td width="134" style="text-align: center">Example</td>
   </tr>
   <tr>
    <td style="text-align: center"><strong>int</strong></td>
    <td>int a</td>
    <td>a = 20</td>
    <td>int a[3]</td>
    <td>a[0] = 10<br>
     a[1] = 20<br>
     a[2] = 30<br>
```

```
        a[3] = &lsquo;\0’</td>
       <td rowspan="2">struct student<br>
        {<br>
        int a;<br>
        char b[10];<br>
        }</td>
       <td rowspan="2">a = 10<br>
        b = &ldquo;Hello&rdquo;</td>
      </tr>
      <tr>
       <td style="text-align: center"><strong>char</strong></td>
       <td>char b</td>
       <td>b=&rsquo;Z&rsquo;</td>
       <td>char b[10]</td>
       <td>b=&rdquo;Hello&rdquo;</td>
      </tr>
    </tbody>
  </table>
</div>
<h4><strong>Below table explains following concepts in C structure.</strong></h4>
<ol>
  <li>How to declare a C structure?</li>
  <li>How to initialize a C structure?</li>
  <li>How to access the members of a C structure?</li>
</ol>
<div>
  <table width="900" border="1" align="center" cellpadding="2">
    <tbody>
      <tr>
       <td style="text-align: center"><strong>Type</strong></td>
       <td style="text-align: center"><strong>Using normal variable</strong></td>
       <td style="text-align: center"><strong>Using pointer variabe</strong></td>
      </tr>
      <tr>
       <td style="text-align: center">Syntax</td>
       <td>struct tag_name<br>
```

```
  {<br>
  data type var_name1;<br>
  data type var_name2;<br>
  data type var_name3;<br>
  };</td>
 <td>struct tag_name<br>
  {<br>
  data type var_name1;<br>
  data type var_name2;<br>
  data type var_name3;<br>
  };</td>
</tr>
<tr>
 <td style="text-align: center">Example</td>
 <td>struct student<br>
  {<br>
  int  mark;<br>
  char name[10];<br>
  float average;<br>
  };</td>
 <td>struct student<br>
  {<br>
  int  mark;<br>
  char name[10];<br>
  float average;<br>
  };</td>
</tr>
<tr>
 <td style="text-align: center">Declaring structure variable</td>
 <td>struct student report;</td>
 <td>struct student *report, rep;</td>
</tr>
<tr>
 <td style="text-align: center">Initializing structure variable</td>
 <td>struct student report = {100, &ldquo;Mani&rdquo;, 99.5};</td>
 <td>struct student rep = {100, &ldquo;Mani&rdquo;, 99.5};<br>
```

```
        report = &amp;amp;rep;</td>
      </tr>
      <tr>
       <td style="text-align: center">Accessing structure members</td>
       <td>report.mark<br>
        report.name<br>
        report.average</td>
       <td>report  -&amp;gt; mark<br>
        report -&amp;gt; name<br>
        report -&amp;gt; average</td>
      </tr>
     </tbody>
    </table>
   </div>
   <h4> <strong>Example program for C structure:</strong></h4>
   <p>This program is used to store and access &amp;ldquo;id, name and percentage&amp;rdquo; for one student. We can also store and access these data for many students using array of structures.</p>
   <div class="program">#include&amp;lt;stdio.h&amp;gt;
#include&amp;lt;string.h&amp;gt;
struct student()
{
   int id;
   char name[20];
   float percentage;
}
int main()
{
   struct student record - {0}; //initializing to null
   record.id-1;
   strcpy(record.name,"Raju");
   record.percentage-86.5;
   printf("Id is: %d \n", record.id);
   printf("Name is: %s \n", record.name);
   printf("Percentage is: %f \n",record.percentage);
   return 0;
}</div><h4>Output:</h4>
<div class="output">Id is: 1
```

Name is: Raju

Percentage is: 86.500000

<h4>Example program – Another way of declaring C structure:</h4>

In this program, structure variable “record” is declared while declaring structure itself. In above structure example program, structure variable “struct student record” is declared inside main function which is after declaring structure.

#include<stdio.h>

#include<string.h>

struct student

{

  int id;

  char name[20];

  float percentage;

}record;

int main()

{

  record.id-1;

  strcpy(record.name, "Nani");

  record.percentage - 86.5;

  printf("Id is: %d \n",record.id);

  printf("Name is: %s \n",record.name);

  printf("Percentage is: %f \n",record.percentage);

  return 0;

} <h4>Output:</h4>

Id is: 1

Name is: Nani

Percentage is: 86.500000

<h4>Example program – Another way of declaring C structure:</h4>

In this program, structure variable “record” is declared while declaring structure itself. In above structure example program, structure variable “struct student record” is declared inside main function which is after declaring structure.

#include<stdio.h>

#include<string.h>

struct student

{

  int id;

  char name[20];

  float percentage;

} record;

```
int main()
{
   record.id-1;
   strcpy(record.name, "Nani");
   record.percentage - 86.5;
   printf("Id is: %d \n", record.id);
   printf("Name is: %s \n", record.name);
   printf("Percentage is: %f \n", record.percentage);
   return 0;
}</div><h4>Output:</h4>
<div class="output">Id is: 1
Name is: Nani
Percentage is: 86.500000</div>
<h4><strong>Uses of C structures:</strong></h4>
<ol>
 <li>C Structures can be used to store huge data. Structures act as a database.</li>
 <li>C Structures can be used to send data to the printer.</li>
 <li>C Structures can interact with keyboard and mouse to store the data.</li>
 <li>C Structures can be used in drawing and floppy formatting.</li>
 <li>C Structures can be used to clear output screen contents.</li>
 <li>C Structures can be used to check computer&rsquo;s memory size etc.</li>
</ol>
 </div>
 <div id="Footer">
            <ul>
            <li><a href="Functions.html"><</a></li>
     <li><a href="PrintfAndScanfFunctions.html">1</a></li>
     <li><a href="DataTypes.html">2</a></li>
     <li><a href="Constants.html">3</a></li>
     <li><a href="Variables.html">4</a></li>
     <li><a href="OperatorsAndExpressions.html">5</a></li>
     <li><a href="DecisionControlStatements.html">6</a></li>
     <li><a href="LoopControlStatements.html">7</a></li>
     <li><a href="CaseControlStatements.html">8</a></li>
     <li><a href="Arrays.html">9</a></li>
     <li><a href="String.html">10</a></li>
```

```
      <li><a href="Pointers.html">11</a></li>
      <li><a href="Functions.html">12</a></li>
      <li><a href="Structures.html">13</a></li>
      <li><a href="#">>></a></li>
    </ul>
  </div>
</div>
</body>
</html>
```

Cascading Style Sheet

CSS.css

```
#Container {
	font-family: "Lucida Grande", "Lucida Sans Unicode", "Lucida Sans", "DejaVu Sans", Verdana, sans-serif;
	height: auto;
	width: 100%;
	margin-top: 0px;
	margin-bottom: 0px;
	border-style: solid;
	border-width: 0px;
	margin-left: 0px;
	margin-right: auto;
	float: none;
}
body {
	margin-left: 0px;
	margin-top: 0px;
	margin-right: 0px;
	margin-bottom: 0px;
	background-color: #F5F5F5;
}
#Header {
	height: 110px;
	width: 900px;
	margin-left: auto;
	margin-right: auto;
	border-width: 0px;
	border-style: solid;
}
#title {
	margin-top:auto;
	margin-bottom:auto;
}
#cssmenu{
	text-decoration: none;
	height: 48px;
```

```
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
		background-color:#300A9F;
}
#Content {
		height: auto;
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
}
#Footer {
		height: 100px;
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
}
#cssmenu,
#cssmenu ul,
#cssmenu ul li,
#cssmenu ul li a {
  padding: 0;
  margin: 0;
  line-height: 1;
  font-family: 'Source Sans Pro', sans-serif;
  font-weight: 500;
  font-size: 16px;
  color: #ffffff;
}
#cssmenu a {
```

```
  text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
}
#cssmenu ul {
  background: #300A9F;
  border-radius: 3px;
  border: 1px solid #2b4479;
  border: 1px solid #2d4373;
  -webkit-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
  -o-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
  -moz-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
  -ms-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
  box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
}
#cssmenu ul > li {
  float: left;
  list-style: none;
}
#cssmenu ul > li > a {
  display: block;
  text-decoration: none;
  padding: 15px 55px;
  position: relative;
}
#cssmenu ul > li > a:hover {
                background: #0C0A73;
                -webkit-box-shadow: inset 0 0 1px #1e2e4f;
                -o-box-shadow: inset 0 0 1px #1e2e4f;
                -moz-box-shadow: inset 0 0 1px #1e2e4f;
                -ms-box-shadow: inset 0 0 1px #1e2e4f;
                box-shadow: inset 0 0 1px #1e2e4f;
                -webkit-transition: all ease .3s;
                -o-transition: all ease .3s;
                -moz-transition: all ease .3s;
                -ms-transition: all ease .3s;
                transition: all ease .3s;
                width: auto;
```

```
}
#cssmenu ul > li > a:hover:before {
  z-index: 2;
  position: absolute;
  border: 1px solid white;
  border-top: 0;
  border-bottom: 0;
  border-right: 0;
  width: 100%;
  height: 100%;
  top: 0;
  left: -1px;
  opacity: .2;
}
#cssmenu ul > li > a:hover:after {
  z-index: 2;
  position: absolute;
  border: 1px solid white;
  border-top: 0;
  border-bottom: 0;
  border-left: 0;
  width: 100%;
  height: 100%;
  top: 0;
  right: -1px;
  opacity: .2;
}
#cssmenu > ul > li > ul {
  opacity: 0;
  visibility: hidden;
  position: absolute;
}
#cssmenu > ul > li:hover > ul {
  opacity: 1;
  visibility: visible;
  position: absolute;
```

```
  border-radius: 0 0 3px 3px;
  -webkit-box-shadow: none;
  -o-box-shadow: none;
  -moz-box-shadow: none;
  -ms-box-shadow: none;
  box-shadow: none;
}
#cssmenu > ul > li > ul {
  width: 200px;
  position: absolute;
}
#cssmenu > ul > li > ul > li {
  float: none;
  position: relative;
}
#cssmenu > ul > li > ul {
  opacity: 0;
  visibility: hidden;
  position: absolute;
}
#Footer ul {
                float:none;

}
#Footer ul > li {
  float: left;
  list-style:none;
  background-color:#E5E5E5;

}
#Footer ul > li > a {
  display: block;
  text-decoration: none;
  padding: 5px 15px;
  position: relative;
  color:#FFFFFF;
```

```
 border:2px 2px 2px 2px;
 border-color:#E7E1E1;
 background-color:#300A9F;
}
#Footer ul > li > a:hover {
        background-color:#370ACF;
        width: auto;
}
#Footer ul > li > a:hover:before {
 position: absolute;
 width: 100%;
 height: 100%;
 top: 0;

}
#Footer ul > li > a: hover: after {
 position: absolute;
 border-bottom-color:#7721E1;
 width: 100%;
 height: 100%;
 top:0;
}
.program{
        white-space:pre;
        margin: 1px 10px 1px;
        display:block;
        font-family: "Courier New", Courier, mono;
  font-size: 12px;
  color: #000000;
  background-color: #FFFFCC;
  padding-top: 8px;
  padding-right: 8px;
  padding-bottom: 8px;
  padding-left: 8px;
  border: #000000;
  border-style: dashed;
```

```
    border-top-width: 1px;
    border-right-width: 1px;
    border-bottom-width: 1px;
    border-left-width: 1px;
    width: auto;
            }
.output{
            white-space:pre;
            margin: 1px 10px 1px;
            display:block;
            font-family: "Courier New", Courier, mono;
    font-size: 12px;
    color: #000000;
    background-color: #CCFFCC;
    padding-top: 8px;
    padding-right: 8px;
    padding-bottom: 8px;
    padding-left: 8px;
    border: #000000;
    border-style: dashed;
    border-top-width: 1px;
    border-right-width: 1px;
    border-bottom-width: 1px;
    border-left-width: 1px;
    width: auto;
            }
.syntax{
            white-space:pre;
            display:block;
            font-family: "Courier New", Courier, mono;
    font-size: 12px;
    color: #000000;
    background-color: #FFCCFF;
    padding-top: 8px;
    padding-right: 8px;
    padding-bottom: 8px;
```

```
padding-left: 8px;

border: #000000;

border-style: dashed;

border-top-width: 1px;

border-right-width: 1px;

border-bottom-width: 1px;

border-left-width: 1px;

width: auto;

        }
```

Registration.css

```
#Container {
	font-family: "Lucida Grande", "Lucida Sans Unicode", "Lucida Sans", "DejaVu Sans", Verdana, sans-serif;
	height: auto;
	width: 100%;
	margin-top: 0px;
	margin-bottom: 0px;
	margin-left: 0;
	margin-right: auto;
	border-style: solid;
	border-width: 0px;
	float: none;
}
body {
	margin-left: 0px;
	margin-top: 0px;
	margin-right: 0px;
	margin-bottom: 0px;
	background-color: #F5F5F5;
}
#Header {
	height: 110px;
	width: 900px;
	margin-left: auto;
	margin-right: auto;
	border-width: 0px;
	border-style: solid;
}
#title {
	margin-top:auto;
	margin-bottom:auto;
}
#cssmenu{
	text-decoration: none;
	height: 50px;
```

```
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
		background-color:#300A9F;
}
#Content {
		height: auto;
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
		font-family: Cambria, "Hoefler Text", "Liberation Serif", Times, "Times New Roman", serif;
		font-size: 18px;
		font-weight: bold;
		text-decoration: none;
		text-align: center;
}
#Footer {
		height: 100px;
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
}
#cssmenu,
#cssmenu ul,
#cssmenu ul li,
#cssmenu ul li a {
  padding: 0;
  margin: 0;
  line-height: 1;
  font-family: 'Source Sans Pro', sans-serif;
```

```
 font-weight: 500;
 font-size: 16px;
 color: #ffffff;

}
#cssmenu a {
 text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
}
#cssmenu ul {
 background: #300A9F;
 border-radius: 3px;
 border: 1px solid #2b4479;
 border: 1px solid #2d4373;
 -webkit-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
 -o-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
 -moz-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
 -ms-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
 box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
}
#cssmenu ul > li {
 float: left;
 list-style: none;
}
#cssmenu ul > li > a {
 display: block;
 text-decoration: none;
 padding: 15px 75px;
 position: relative;
}
#cssmenu ul > li > a:hover {
        background: #0C0A73;
        -webkit-box-shadow: inset 0 0 1px #1e2e4f;
        -o-box-shadow: inset 0 0 1px #1e2e4f;
        -moz-box-shadow: inset 0 0 1px #1e2e4f;
        -ms-box-shadow: inset 0 0 1px #1e2e4f;
        box-shadow: inset 0 0 1px #1e2e4f;
```

```
            -webkit-transition: all ease .3s;
            -o-transition: all ease .3s;
            -moz-transition: all ease .3s;
            -ms-transition: all ease .3s;
            transition: all ease .3s;
            width: auto;
}
#cssmenu ul > li > a:hover:before {
  z-index: 2;
  position: absolute;
  border: 1px solid white;
  border-top: 0;
  border-bottom: 0;
  border-right: 0;
  width: 100%;
  height: 100%;
  top: 0;
  left: -1px;
  opacity: .2;
}
#cssmenu ul > li > a:hover:after {
  z-index: 2;
  position: absolute;
  border: 1px solid white;
  border-top: 0;
  border-bottom: 0;
  border-left: 0;
  width: 100%;
  height: 100%;
  top: 0;
  right: -1px;
  opacity: .2;
}
#cssmenu > ul > li > ul {
  opacity: 0;
  visibility: hidden;
```

```
 position: absolute;
}
#cssmenu > ul > li:hover > ul {
 opacity: 1;
 visibility: visible;
 position: absolute;
 border-radius: 0 0 3px 3px;
 -webkit-box-shadow: none;
 -o-box-shadow: none;
 -moz-box-shadow: none;
 -ms-box-shadow: none;
 box-shadow: none;
}
#cssmenu > ul > li > ul {
 width: 200px;
 position: absolute;
}
#cssmenu > ul > li > ul > li {
 float: none;
 position: relative;
}
#cssmenu > ul > li > ul {
 opacity: 0;
 visibility: hidden;
 position: absolute;
}
h5 {
        font-size: 36px;
        font-family: Cambria, "Hoefler Text", "Liberation Serif", "Times", "Times New Roman", "serif";
        font-style: normal;
        font-weight: bold;
}
#subtitle{
        float: center;
        text-align: center;
        font-family: Baskerville, "Palatino Linotype", Palatino, "Century Schoolbook L", "Times New Roman", serif;
```

```
        font-size: 40px;
        font-style: normal;
        font-weight: bold;
        font-variant: normal;
        line-height: normal;
        text-transform: uppercase;
        color: #2C0383;
}
a:link {
        color: #0C5098;
        text-decoration: none;
}
a:visited {
        text-decoration: none;
        color: #E90404;
}
a:hover {
        text-decoration: underline;
        color: #081BF7;
}
a:active {
        text-decoration: none;
        color: #16BF1A;
}
.textbar {
        padding: 6px 6px 6px 6px;

}
.buttons {
        padding: 6px 15px 6px 15px;
}
```

Contact.css

```
#Container {
	font-family: "Lucida Grande", "Lucida Sans Unicode", "Lucida Sans", "DejaVu Sans", Verdana, sans-serif;
	height: auto;
	width: 100%;
	margin-top: 0px;
	margin-bottom: 0px;
	border-style: solid;
	border-width: 0px;
	margin-left: 0px;
	margin-right: auto;
	float: none;
}
body {
	margin-left: 0px;
	margin-top: 0px;
	margin-right: 0px;
	margin-bottom: 0px;
	background-color: #F5F5F5;
}
#Header {
	height: 110px;
	width: 900px;
	margin-left: auto;
	margin-right: auto;
	border-width: 0px;
	border-style: solid;
}
#title {
	margin-top:auto;
	margin-bottom:auto;
}
#cssmenu{
	text-decoration: none;
	height: 50px;
```

```
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
		background-color:#300A9F;
}
#Content {
		height: auto;
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
		text-align: center;
}
#Footer {
		height: 100px;
		width: 100%;
		margin-right: auto;
		margin-left: auto;
		border-width: 0px;
		border-style: solid;
}
#cssmenu,
#cssmenu ul,
#cssmenu ul li,
#cssmenu ul li a {
  padding: 0;
  margin: 0;
  line-height: 1;
  font-family: 'Source Sans Pro', sans-serif;
  font-weight: 500;
  font-size: 16px;
  color: #ffffff;
}
```

```
#cssmenu a {
  text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
}
#cssmenu ul {
  background: #300A9F;
  border-radius: 3px;
  border: 1px solid #2b4479;
  border: 1px solid #2d4373;
  -webkit-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
  -o-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
  -moz-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
  -ms-box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
  box-shadow: 0 1px 2px rgba(2, 2, 2, 0.25), inset 0 1px 1px rgba(255, 255, 255, 0.15);
}
#cssmenu ul > li {
  float: left;
  list-style: none;
}
#cssmenu ul > li > a {
  display: block;
  text-decoration: none;
  padding: 15px 75px;
  position: relative;
}
#cssmenu ul > li > a:hover {
                background: #0C0A73;
                -webkit-box-shadow: inset 0 0 1px #1e2e4f;
                -o-box-shadow: inset 0 0 1px #1e2e4f;
                -moz-box-shadow: inset 0 0 1px #1e2e4f;
                -ms-box-shadow: inset 0 0 1px #1e2e4f;
                box-shadow: inset 0 0 1px #1e2e4f;
                -webkit-transition: all ease .3s;
                -o-transition: all ease .3s;
                -moz-transition: all ease .3s;
                -ms-transition: all ease .3s;
                transition: all ease .3s;
```

```
            width: auto;
}
#cssmenu ul > li > a:hover:before {
  z-index: 2;
  position: absolute;
  border: 1px solid white;
  border-top: 0;
  border-bottom: 0;
  border-right: 0;
  width: 100%;
  height: 100%;
  top: 0;
  left: -1px;
  opacity: .2;
}
#cssmenu ul > li > a:hover:after {
  z-index: 2;
  position: absolute;
  border: 1px solid white;
  border-top: 0;
  border-bottom: 0;
  border-left: 0;
  width: 100%;
  height: 100%;
  top: 0;
  right: -1px;
  opacity: .2;
}
#cssmenu > ul > li > ul {
  opacity: 0;
  visibility: hidden;
  position: absolute;
}
#cssmenu > ul > li:hover > ul {
  opacity: 1;
  visibility: visible;
```

```
 position: absolute;
 border-radius: 0 0 3px 3px;
 -webkit-box-shadow: none;
 -o-box-shadow: none;
 -moz-box-shadow: none;
 -ms-box-shadow: none;
 box-shadow: none;
}
#cssmenu > ul > li > ul {
 width: 200px;
 position: absolute;
}
#cssmenu > ul > li > ul > li {
        float: none;
        position: relative;
        font-family: Constantia, "Lucida Bright", "DejaVu Serif", Georgia, serif;
}
#cssmenu > ul > li > ul {
 opacity: 0;
 visibility: hidden;
 position: absolute;
}
#subtitle{
        float: center;
        text-align: center;
        font-family: Baskerville, "Palatino Linotype", Palatino, "Century Schoolbook L", "Times New Roman", serif;
        font-size: 40px;
        font-style: normal;
        font-weight: bold;
        font-variant: normal;
        line-height: normal;
        text-transform: uppercase;
        color: #2C0383;
}
```

TESTING

Testing Fundamentals

Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing represents an interesting anomaly for the software. During earlier definition and development phases, it was attempted to build software from an abstract concept to a tangible implementation. No system is error free because it is so till the next error drops up during any phase of the development or usage of the product. A sincere effort however needs to be put to bring out a product that is satisfactory. The testing phase involves the testing of development system using various data. Preparation of the test data plays a vital role in system testing. After preparing the test data, the system under study was tested using those data. While testing the system, by using the test data, errors were found and corrected by using the following testing steps and corrections were also noted for future use. Thus, a series of testing is performed on the proposed system before the system is ready for implementation. Software testing is more than just error detection.

Testing software is operating the software under controlled conditions, to (1) verify that it behaves “as specified” (2) to detect errors, and (3) to validate that what has been specified is what the user actually wanted.

In general, Software Testing is a combination of the following activities which are inter-related with one another.

- Error Detection
- Verification
- Validation

Hence the purpose of testing is verification, validation and error detection in order to find problems and the purpose of finding those problems is to get them fixed.

Test Cases:

The importance of software testing and its implications cannot be overemphasized. Software testing is a critical element of Software Quality Assurance and represents the ultimate review of the specifications, design and coding.

Software Testing:

As the coding is completed according to the requirement we have to test the quality of the software. Software testing is a critical element of the software quality assurance and represents the ultimate review of specification, design and coding. Although testing is to uncover the errors in the software functions appear to be working as per the specification, those performance requirements appear to have been met. In addition, data collected as testing is conducted provide a good indication of software reliability and some indications of software quality as a whole. To assure the software quality we conduct both white box testing and black box testing.

White box testing:

White box testing is a test case design method that uses the control structure of the procedural designs to derive test cases. As we are using a non-procedural language, there is very small scope for the white box testing. Whenever it is necessary, there the control structures are tested and successfully passed all the control structures with a very minimum errors.

Guarantee that all independent paths within a module have been exercised at least once

- Exercise all logical decisions on their true and false sides
- Exercise all loops their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.

Black box testing:

It focuses on the functional requirements to the software. It enables to derive sets of input conditions that will fully exercise all functional requirements for a program. The Black box testing finds almost all errors. It finds some interface errors and errors in accessing the database and some performance errors. In Black box testing we use two techniques equivalence partitioning the boundary volume analyzing technique.

Black box testing attempts to find errors in the following categories.

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access.
- Performance errors.
- Initialization and termination errors.

Unlike White box testing, which is performed earlier in the testing process, Black box testing tends to applied during later stages of testing.

Testing Strategies:

It is designated to uncover weakness that was not detected in the earlier tests. The total system is tested for recovery and fallback after various major failures to ensure that no data are lost. An acceptance test is done to validity and reliability of the system. The philosophy behind the testing is to find error

in project. There are many test cases designed with this in model. The flow of testing is as follows.

Code Testing:

Specification testing is done to check if the program does with it should do and how it should behave under various condition or combinations and submitted for processing in the system and it is checked if any overlaps occur during the processing. This strategy examines the logic of the program. Here only syntax of the code is tested. In code testing syntax errors are corrected, to ensure that the code is perfect.

Unit Testing:

The first level of testing is called unit testing. Here different modules are tested against the specification produced running the design of the modules. Unit testing is done to test the working of individual modules with test oracles. Unit testing comprises a set of tests performed by an individual programmer prior to integration of the units into a large system. A program unit is usually small enough that the programmer who developed it can test it in a great detail. Unit testing focuses first on the modules to locate errors. These errors are verified and corrected so that the unit perfectly fits to the project.

System Testing:

The next level of testing is system testing and acceptance testing. This testing is done to check if the system has met its requirements and to find the external behavior of the system. System testing involves two kinds of activities.

- Integration testing
- Acceptance testing

- Integration Testing:

The next level of testing is called the Integration testing. In this many tested modules are combined into subsystems, which were then tested. Test case data is prepared to check the control flow of all the modules and to exhaust all possible inputs to the program. Situations like treating the modules when there is no data entered in the test box is also tested.

This testing strategy dictates the order in which modules must be available, and exerts strong influence on the order in which the modules must be written, debugged and unit tested. In integration testing, all modules on which unit testing is performed are integrated together and tested.

- Acceptance Testing:

This testing is performed finally by user to demonstrate that the implemented system satisfies its requirements. The user gives various inputs to get required outputs.

Specification Testing:

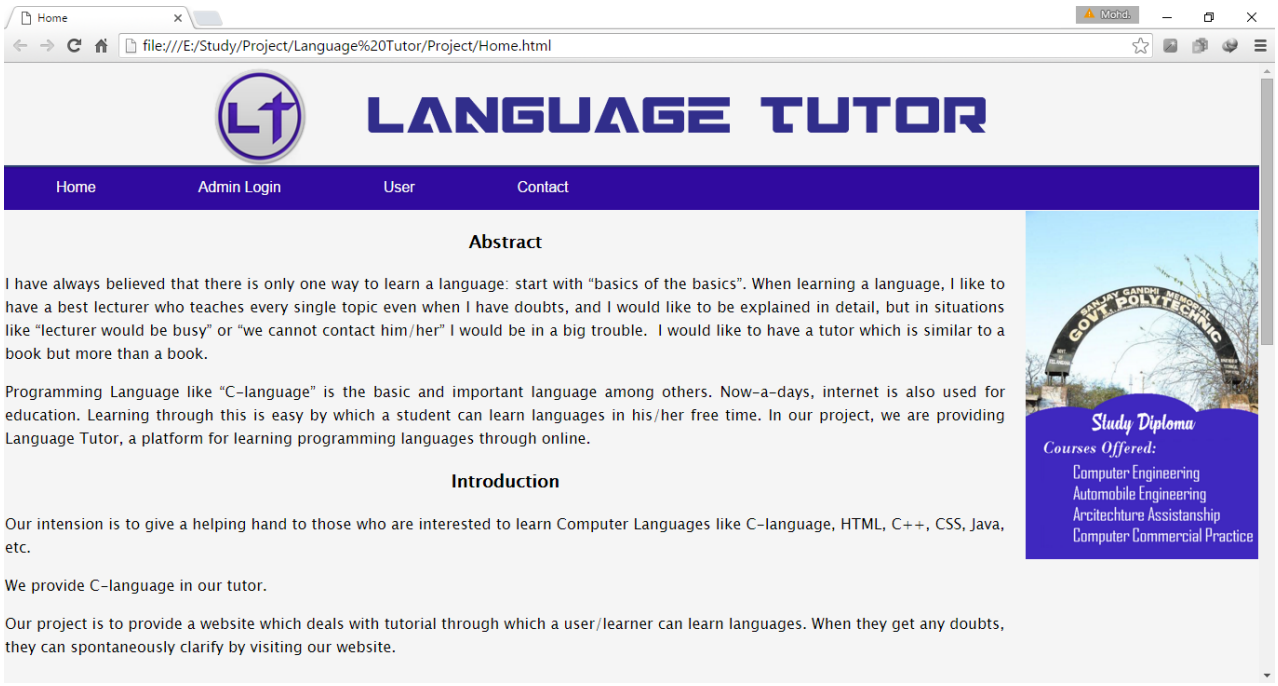
This is done to check if the program does what it should do and how it should behave under various conditions or combination and submitted for processing in the system and it is checked if any overlaps occur during the processing.

Performance Time Testing:

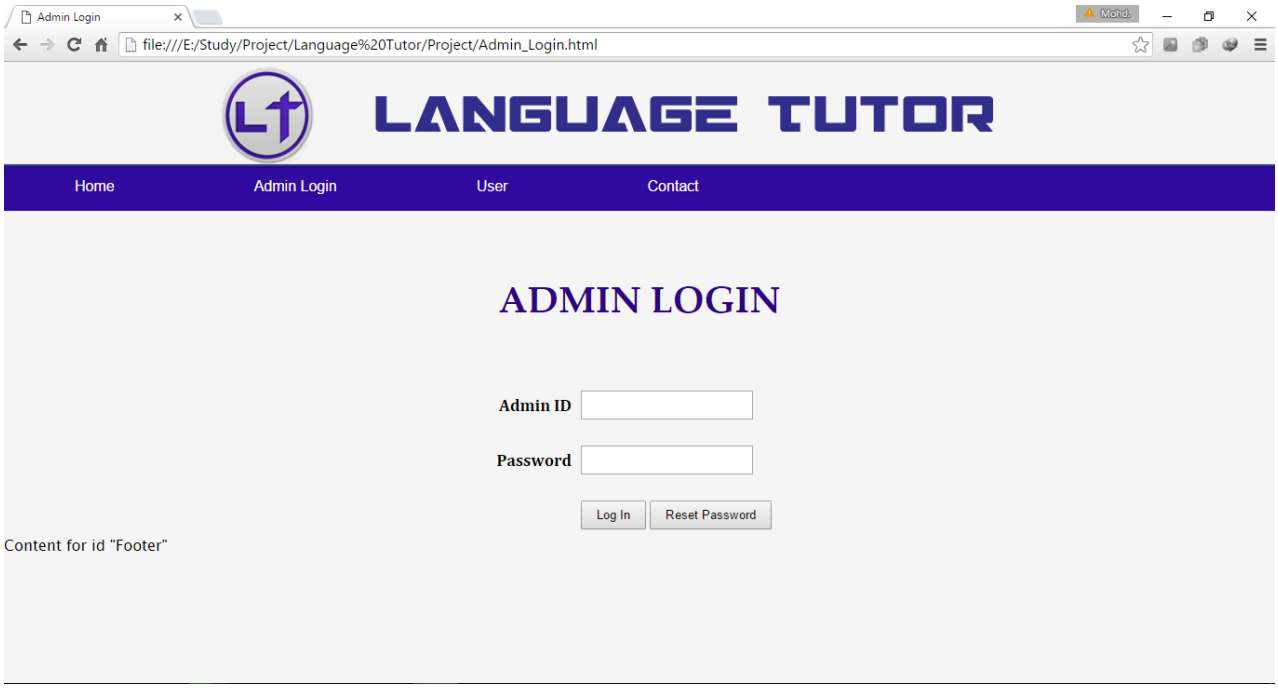
This is done to determine how long it takes to accept and respond i.e., the total time for processing when it has to handle quite a large number of records. It is essential to check the exception speed of the system, which runs well with only a handful of test transactions. Such systems might be slow when fully loaded. So testing is done by providing large number of data for processing. A system testing is designed to uncover weaknesses that were not detected in the earlier tests. The total system is tested for recovery and fall back after various major failures to ensure that no data is lost during an emergency, an acceptance test is done to ensure about the validity and reliability of the system.

Screenshots

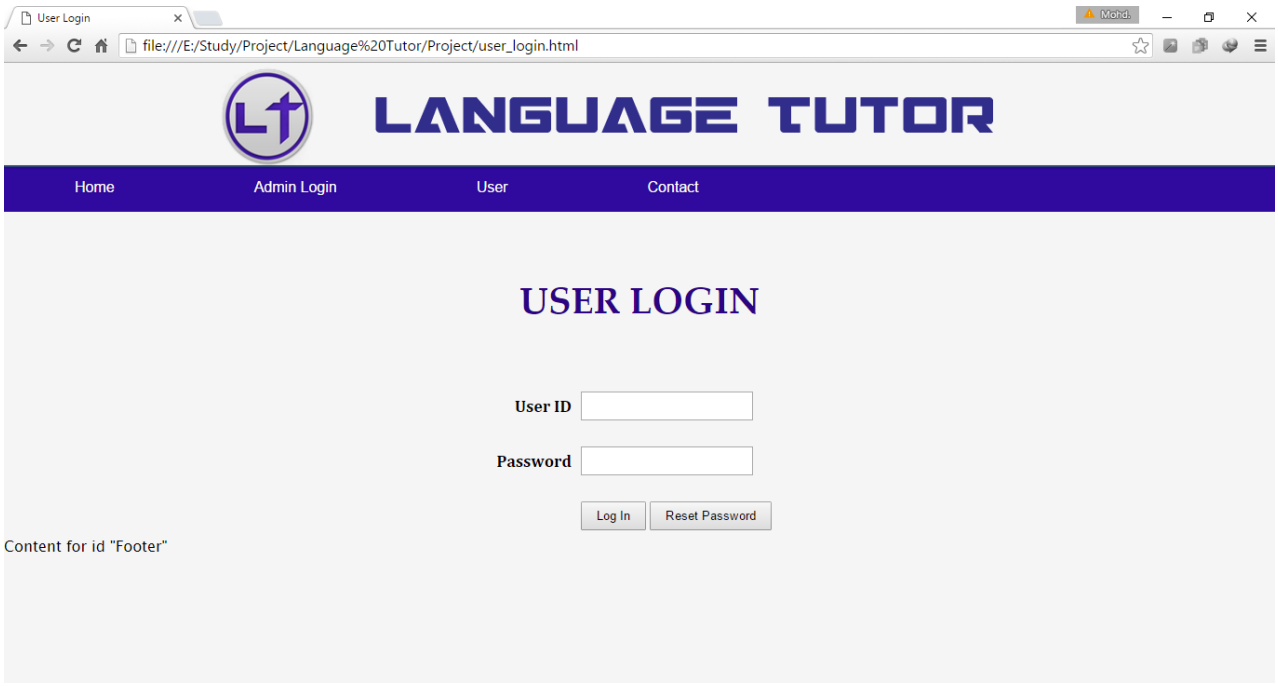
Home.html



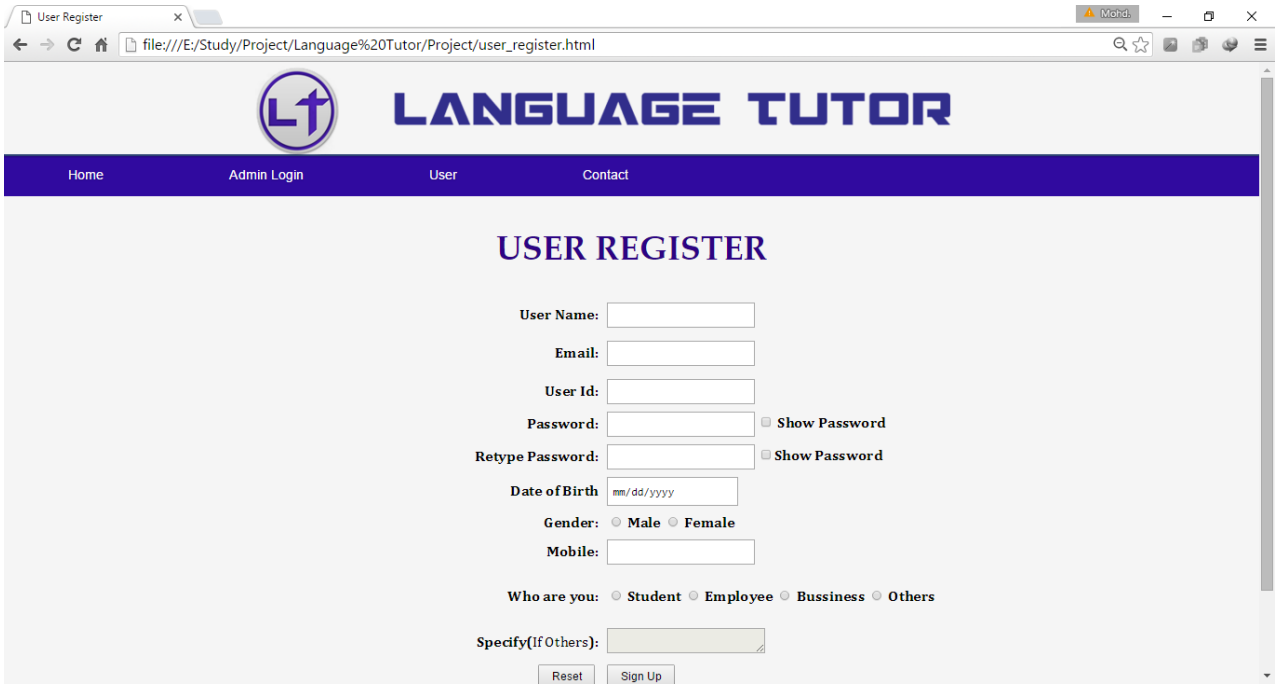
Admin_Login.html



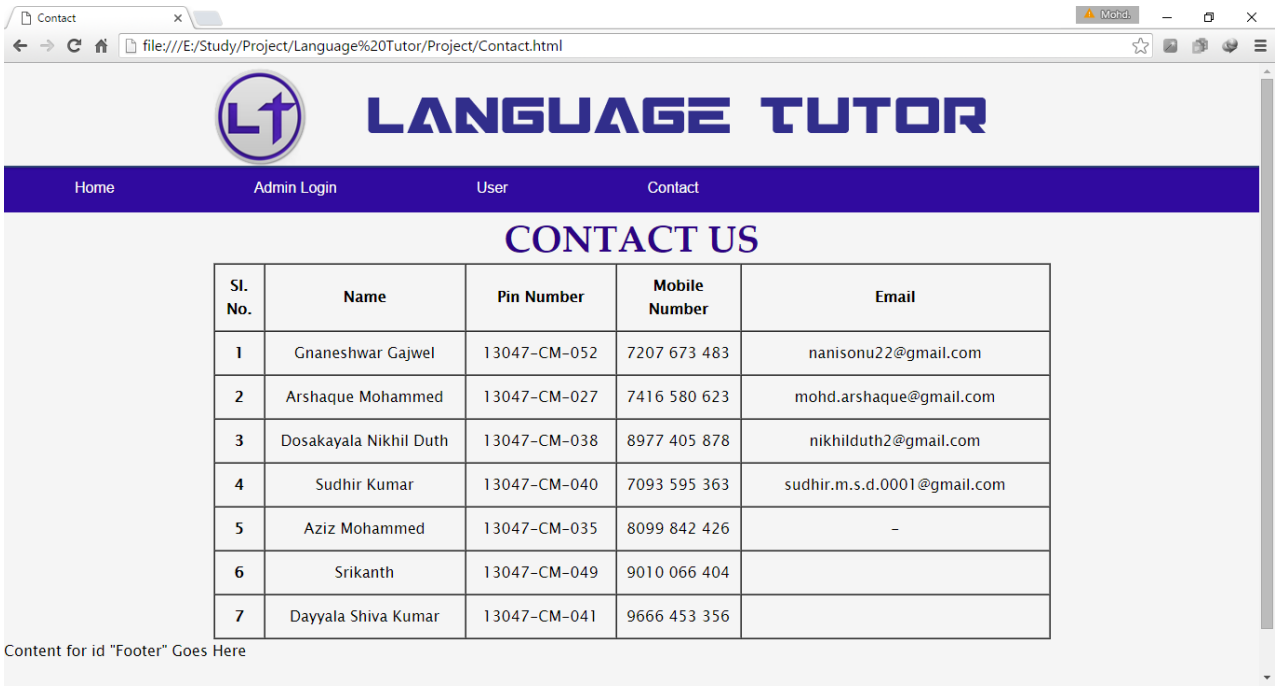
User_Login.html



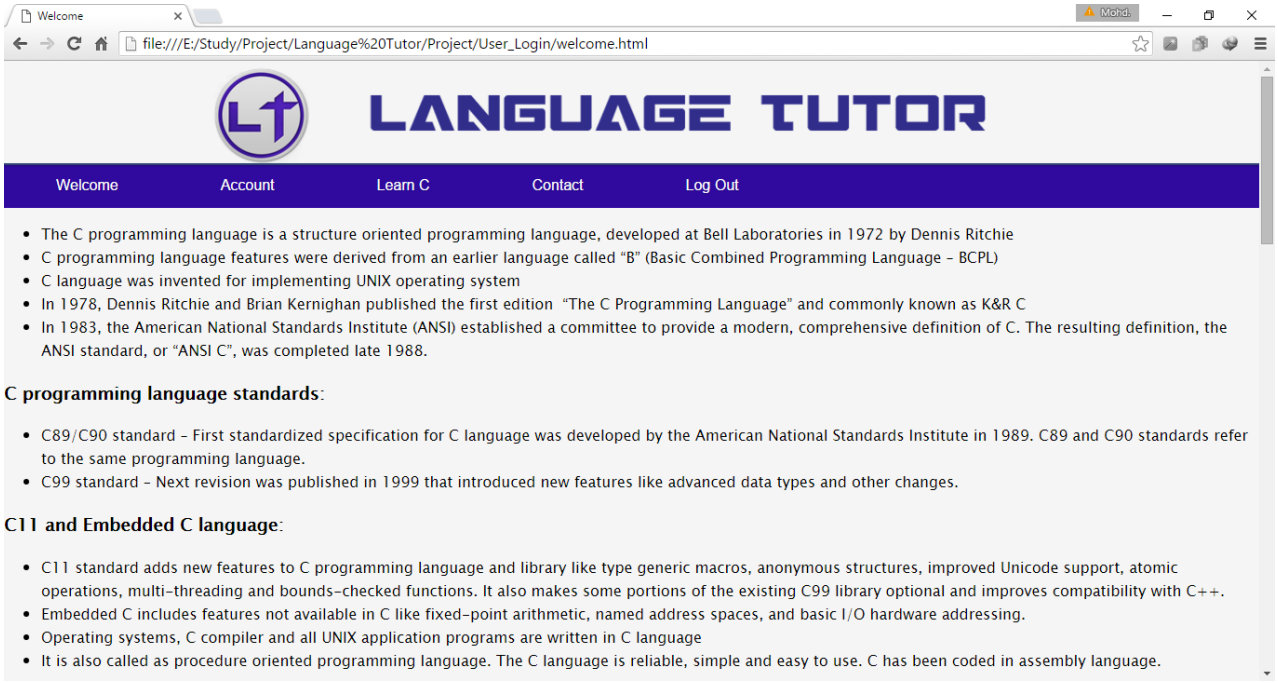
User_Register.html



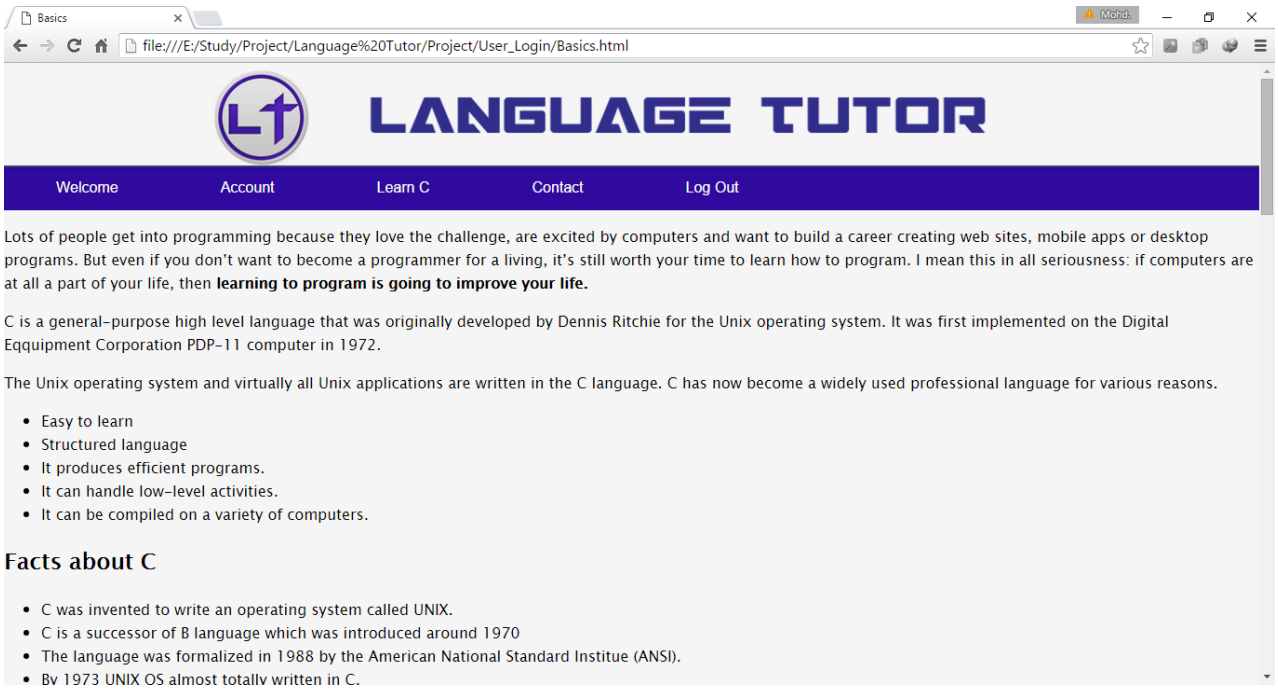
Contact.html



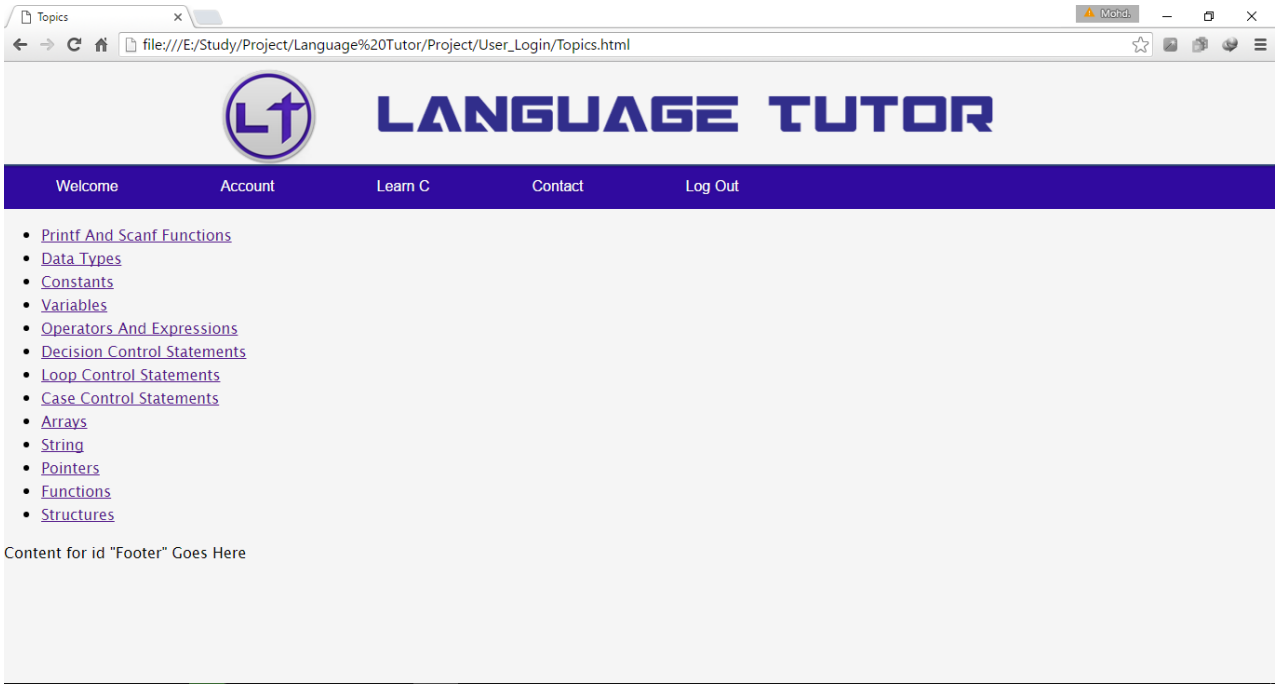
Welcome.html



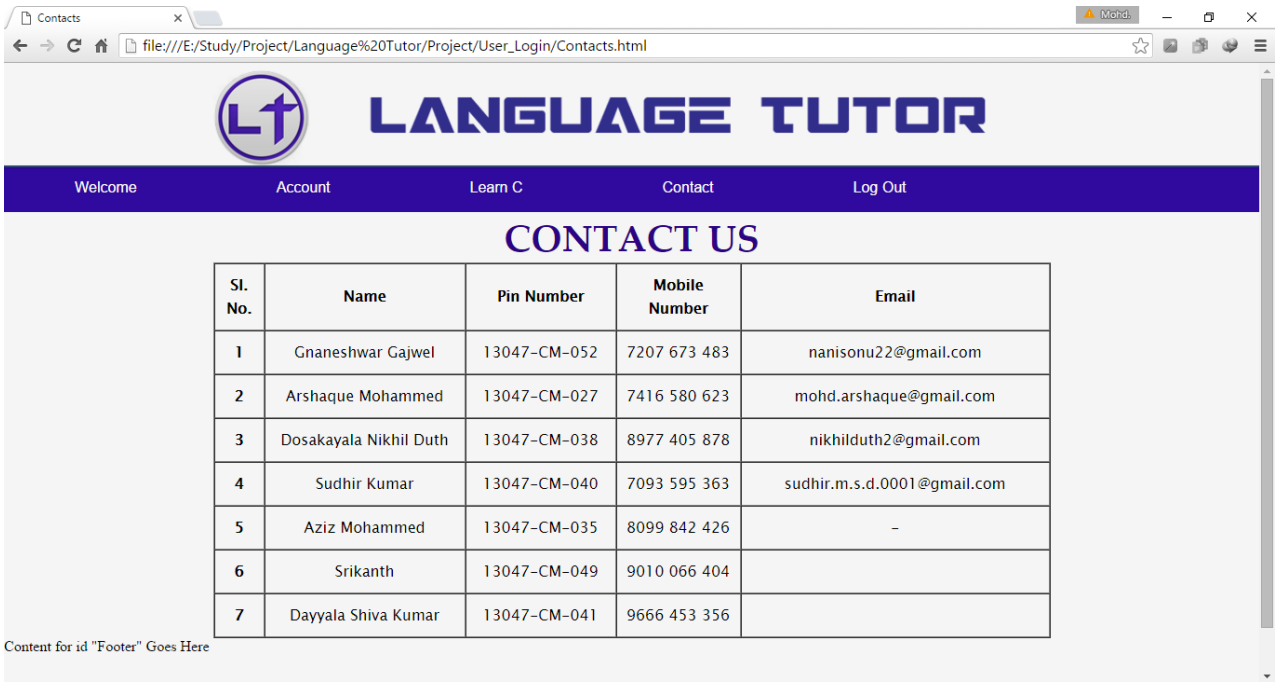
Basics.html



Topics.html

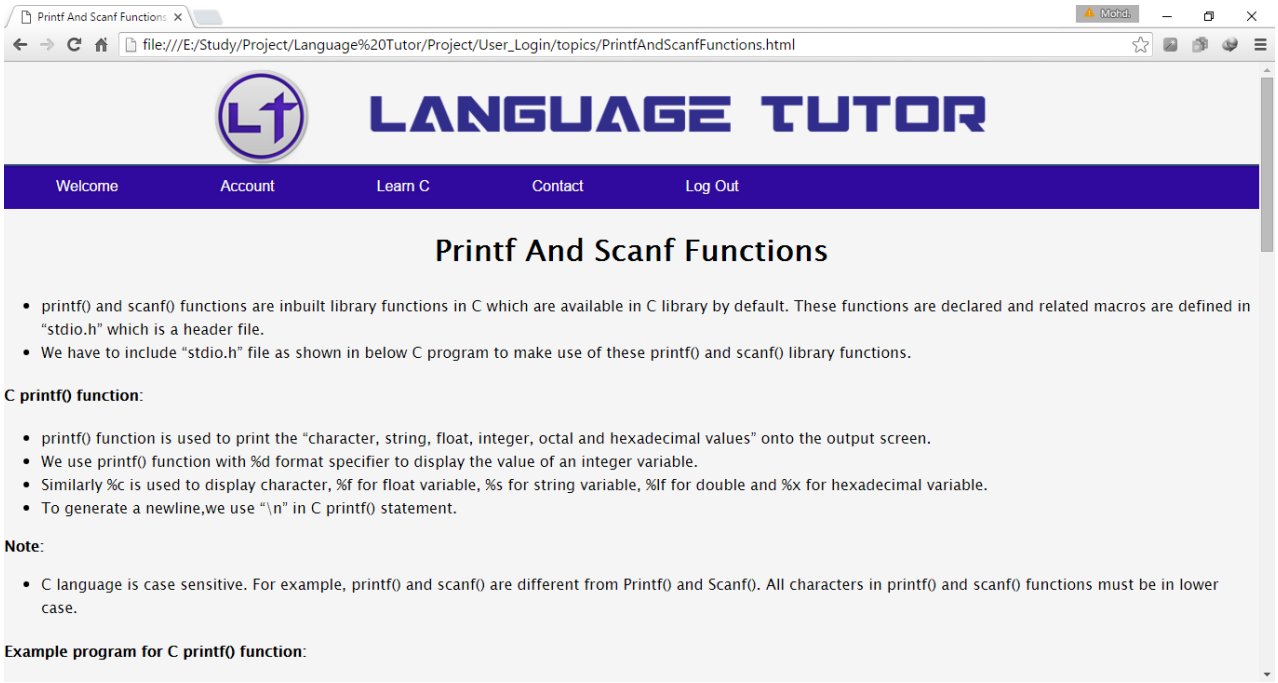


Contacts.html

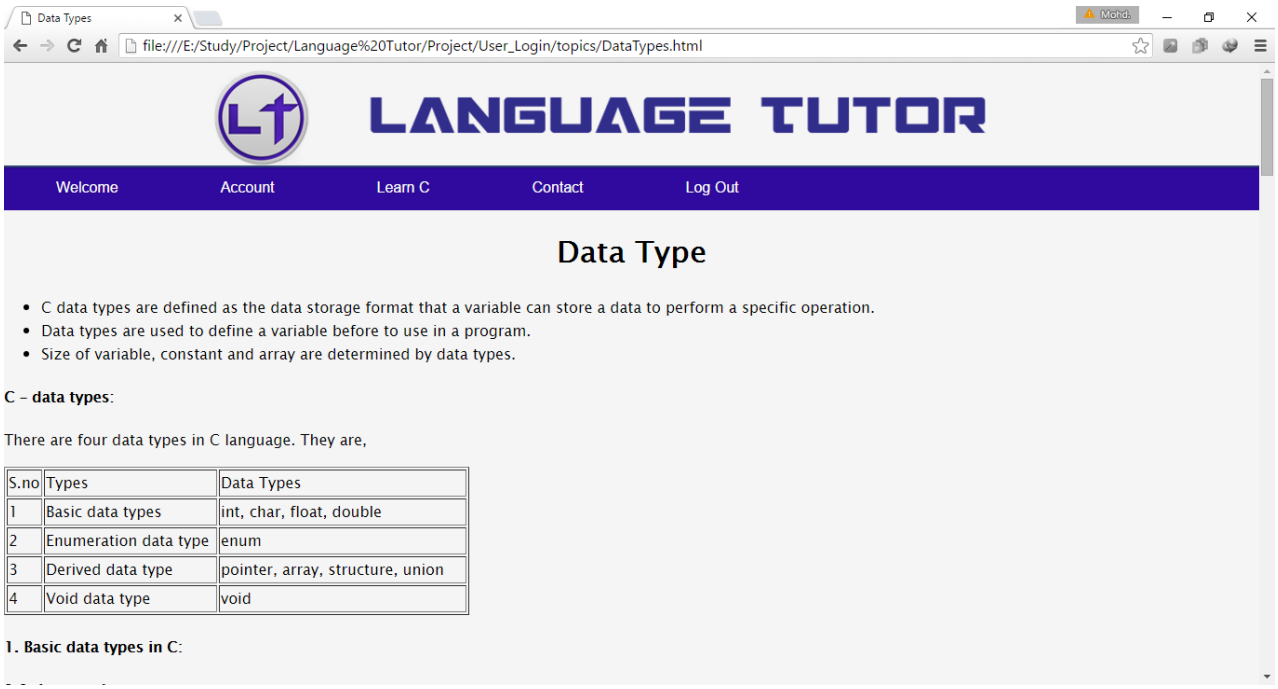


Sl. No.	Name	Pin Number	Mobile Number	Email
1	Gnaneshwar Gajwel	13047-CM-052	7207 673 483	nanisonu22@gmail.com
2	Arshaque Mohammed	13047-CM-027	7416 580 623	mohd.arshaque@gmail.com
3	Dosakayala Nikhil Duth	13047-CM-038	8977 405 878	nikhilduth2@gmail.com
4	Sudhir Kumar	13047-CM-040	7093 595 363	sudhir.m.s.d.0001@gmail.com
5	Aziz Mohammed	13047-CM-035	8099 842 426	-
6	Srikanth	13047-CM-049	9010 066 404	
7	Dayyala Shiva Kumar	13047-CM-041	9666 453 356	

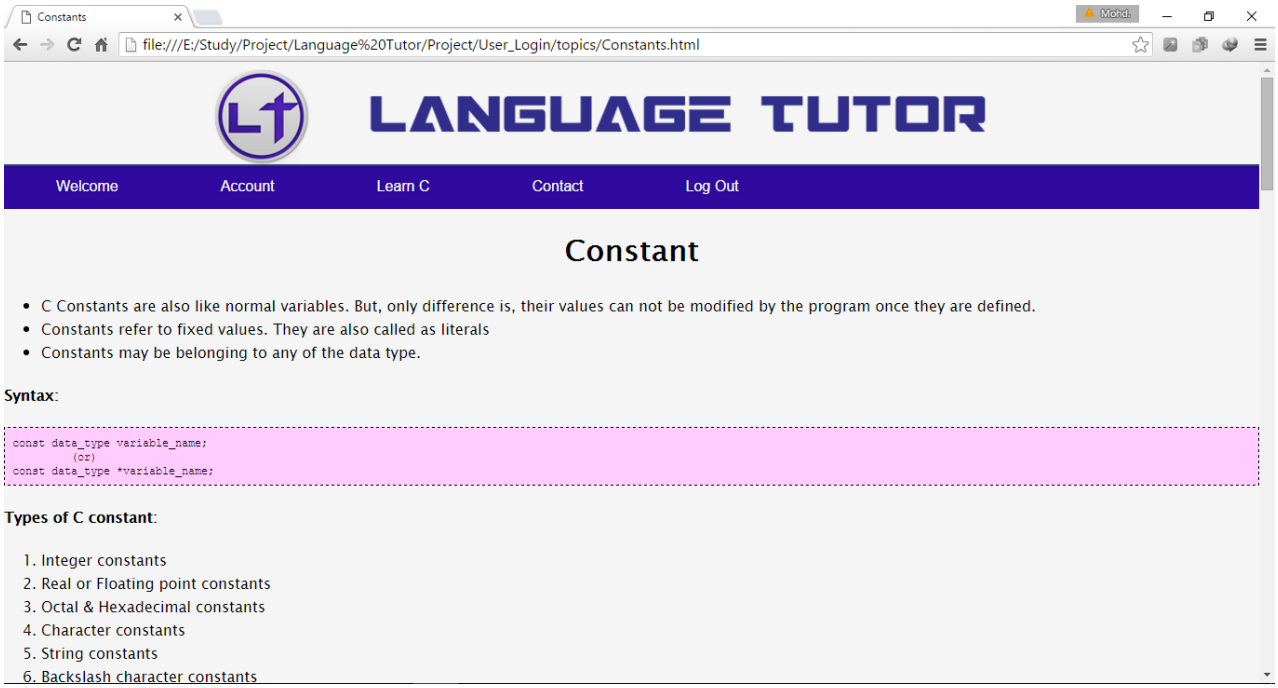
PrintfAndScanfFunctions.html



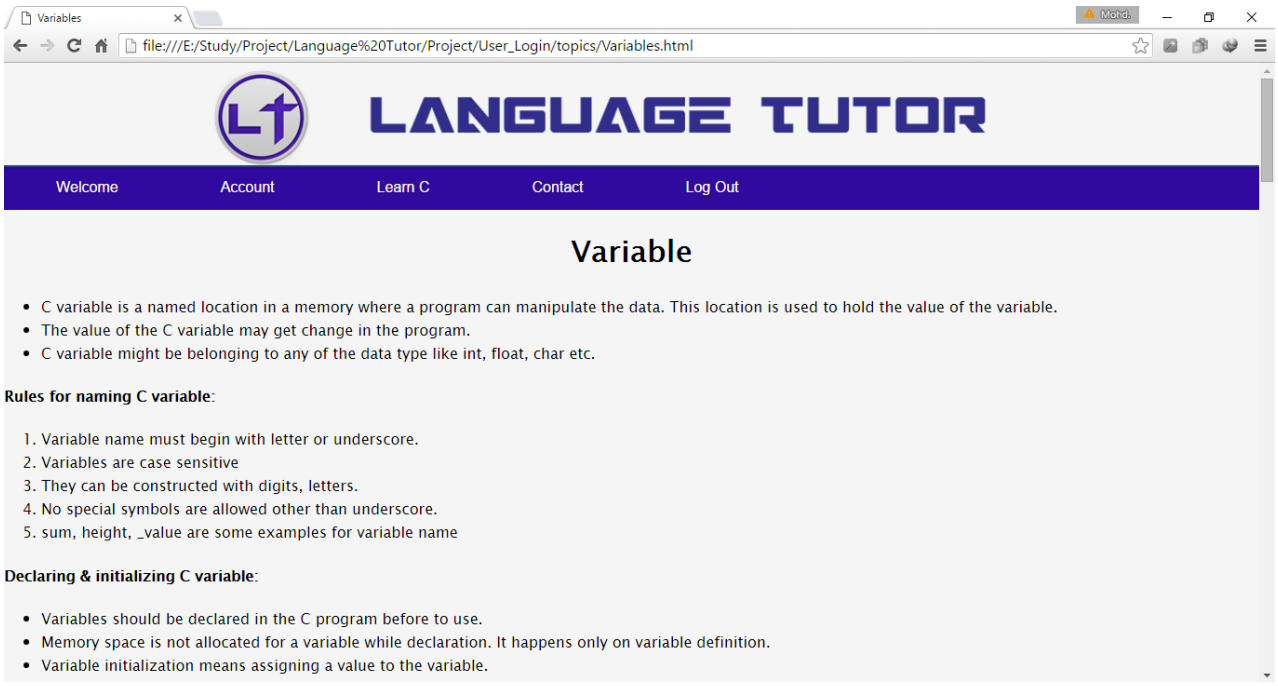
DataTypes.html



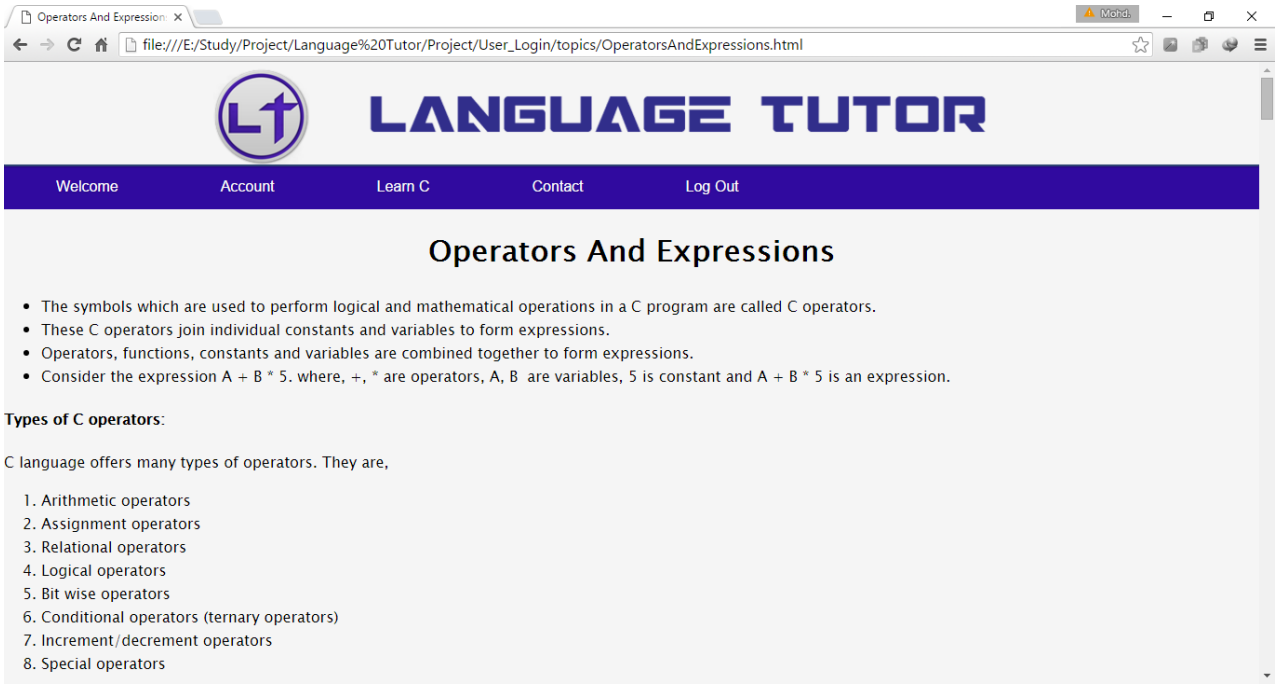
Constants.html



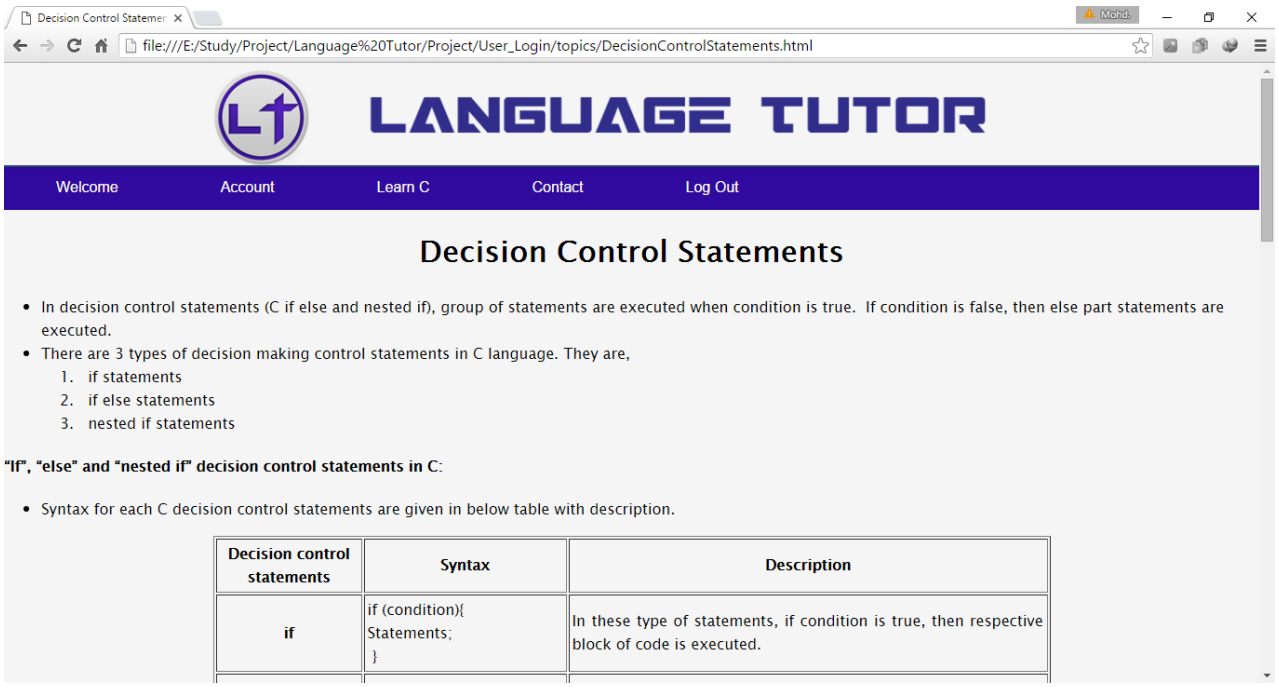
Variables.html



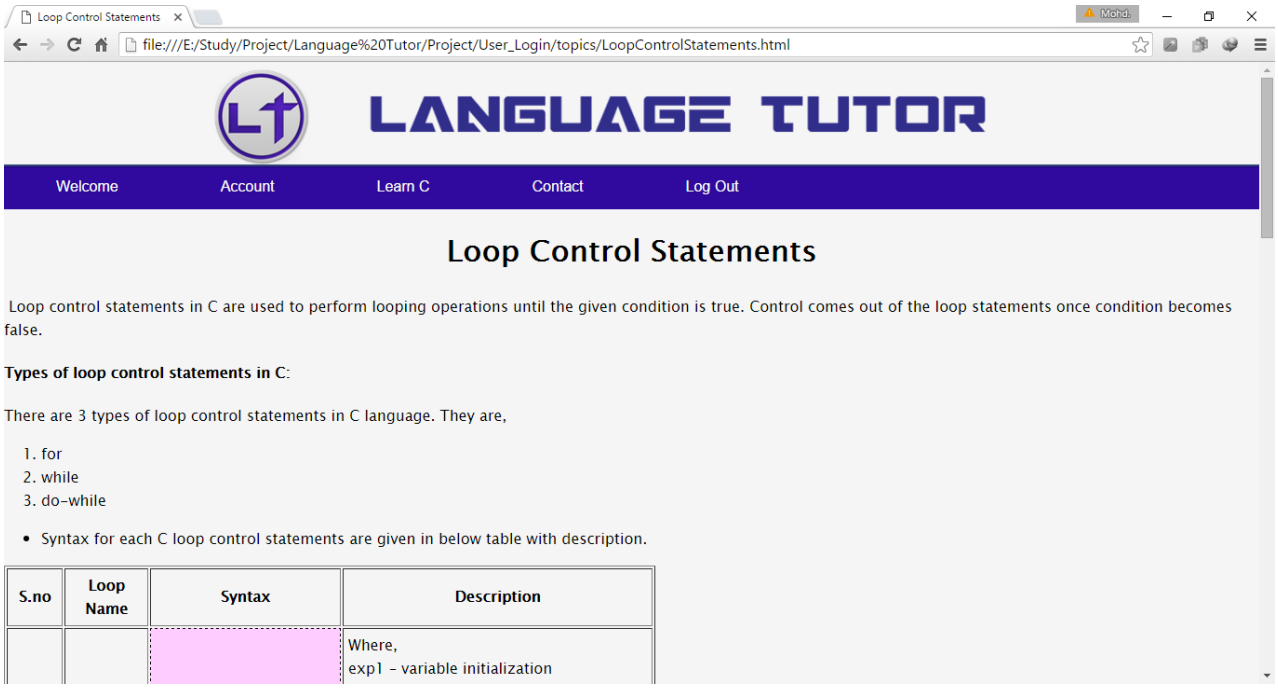
OperatorsAndExpressions.html



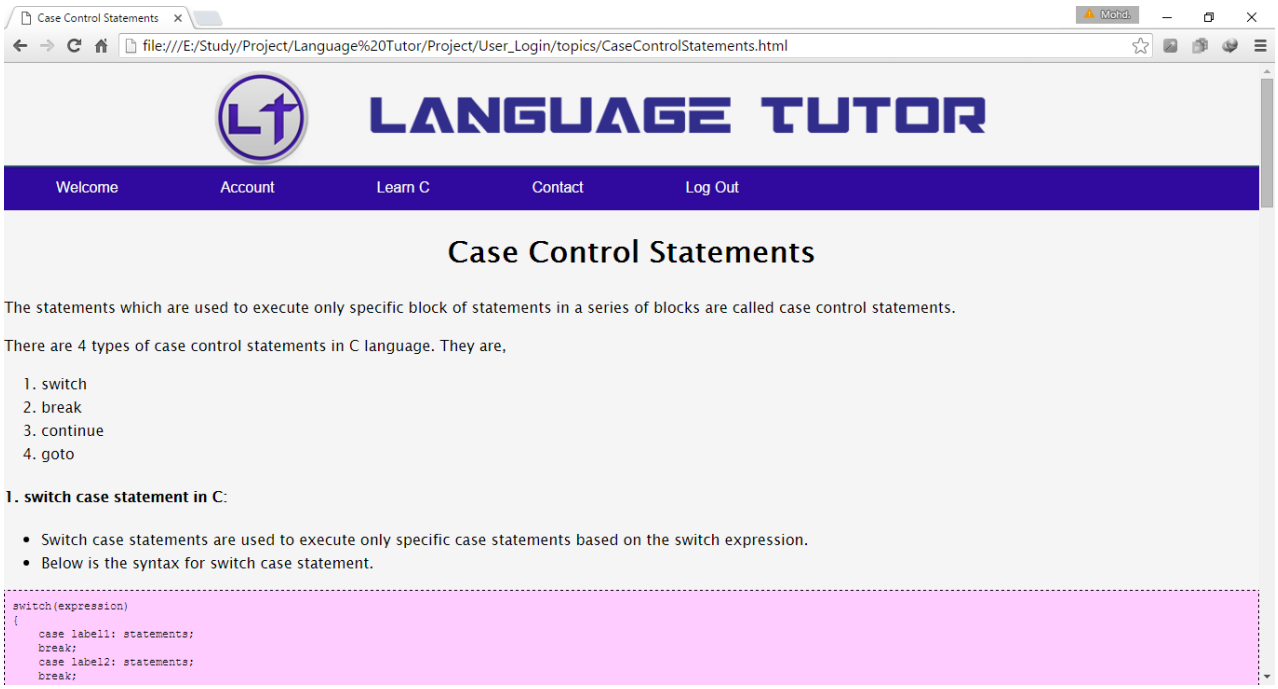
DecisionControlStatements.html



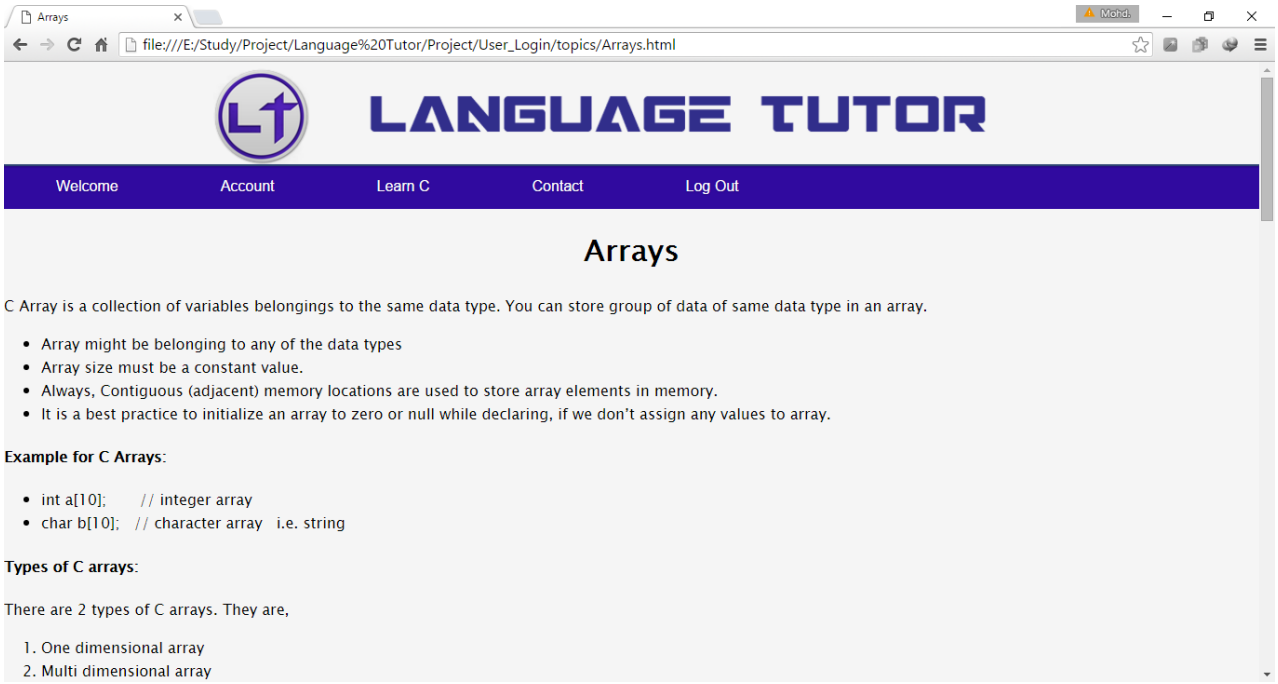
LoopControlStatements.html



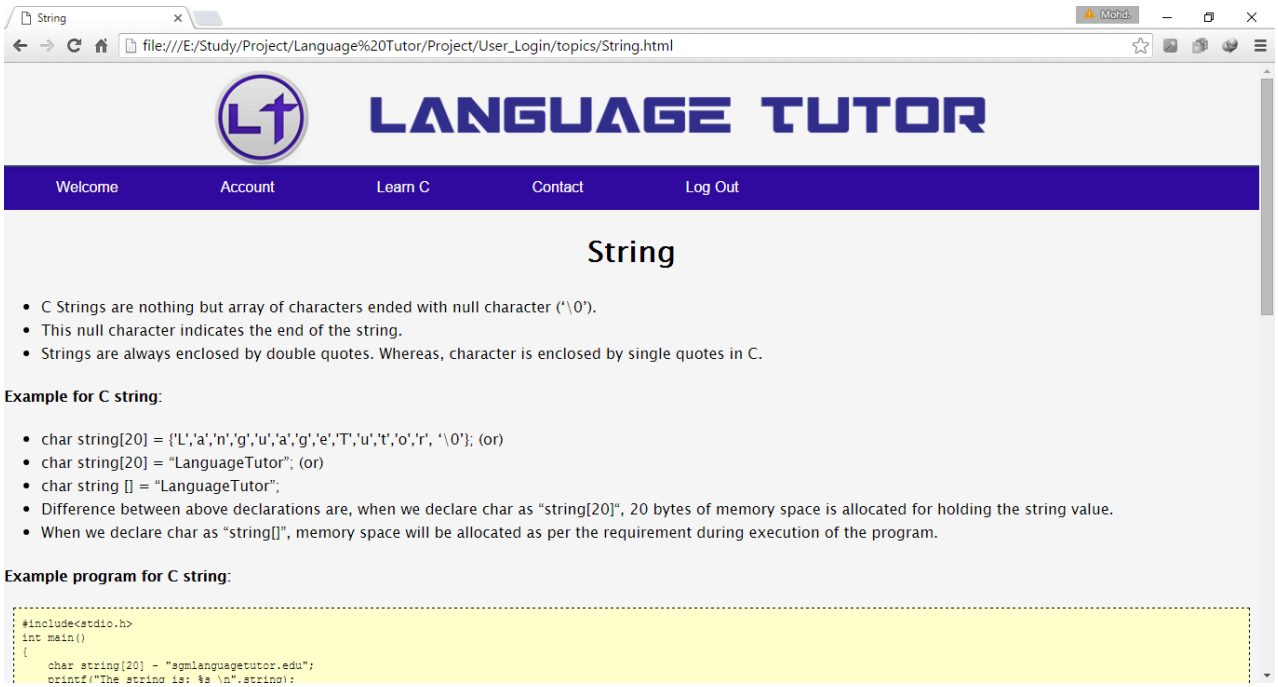
CaseControlStatements.html



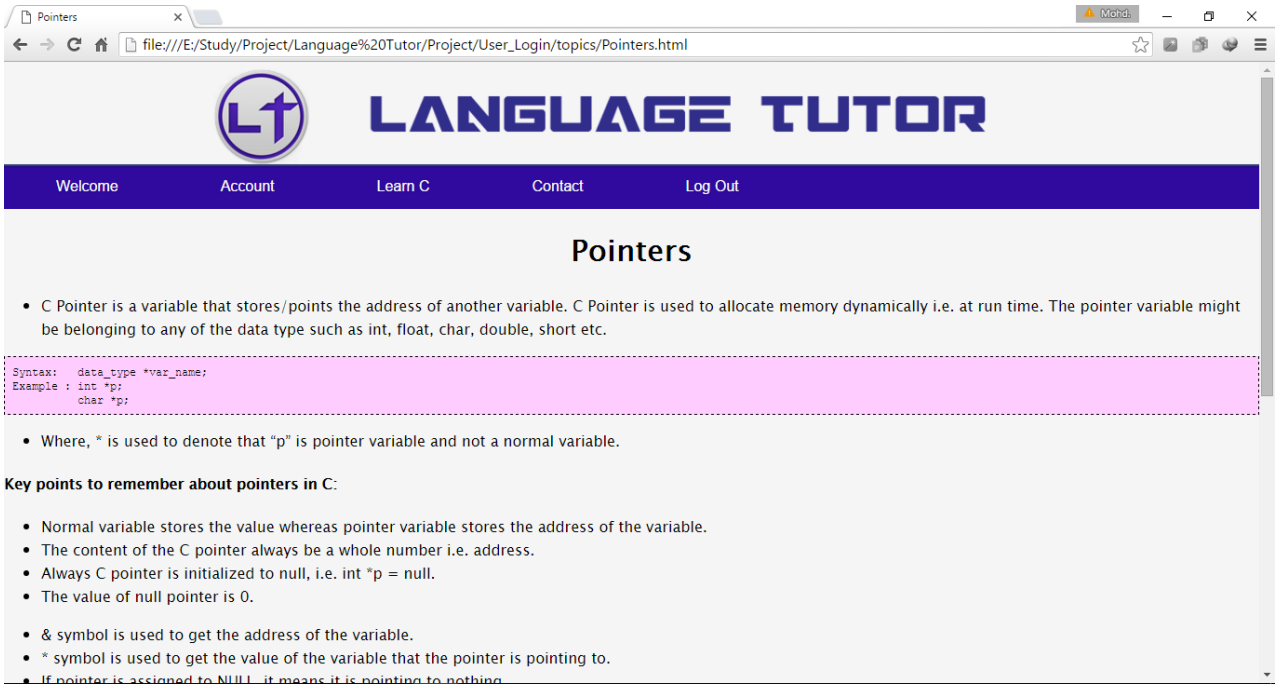
Arrays.html



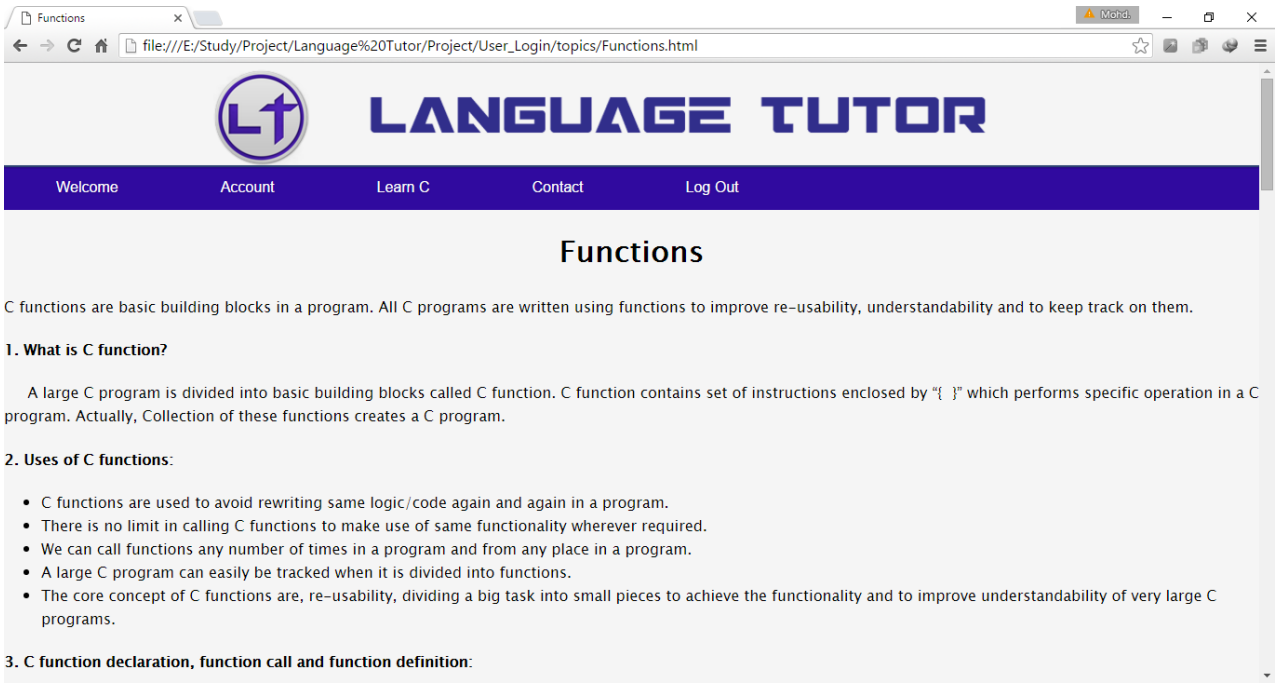
String.html



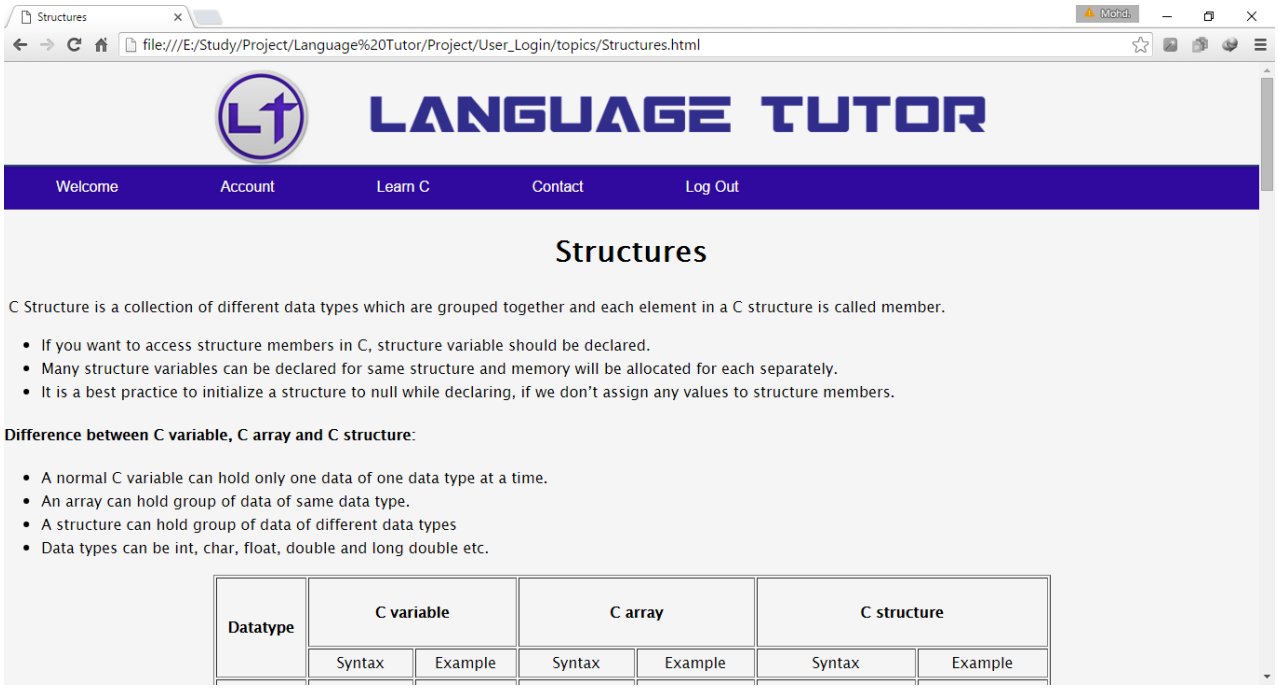
Pointers.html



Functions.html



Structures.html



CONCLUSION

Conclusion

This project has been a rewarding experience in more than one way. The entire project work has enlightened us in the following areas.

1. We have gained an insight into the working of the projects. This represents a typical real world situation.
2. Our understanding of database design has been strengthened. This is because in order to generate the final reports of database designing has to be properly followed.
3. Scheduling a project and adhering to that schedule creates a strong sense of time management.
4. Sense of teamwork has developed and confidence of handling real life project has increased to a great extent.
5. Initially, there were problems with the time management and group co-ordination. But we succeeded when we realize the strength of team work.
6. There were so many problems, we team members solved the problems with help of our guide.

BIBLIOGRAPHY

Bibliography

References made from:

Books Referred:

1. C-09 Curriculum (Diploma in Computer Engineering CME)
2. The Complete Reference C
   - Balaguruswamy

Sites Referred:

1. www.w3schools.com
2. www.youtube.com