IWPCUG

Wed April 5th 2017

A Dabblers View of the Evolution of Programming

by

Mike Hoar

In the Beginning was the MACHINE

But how do you tell it what to do?

The answer is Machine Code.
And every machine has its own set of instruction codes.

These instructions were in numerical code occupying 1 byte, and meaningless to another person.

That is why **Assembly Code** was invented. 2 character codes which are ‘translated into 1, or a short series of, machine code instructions.

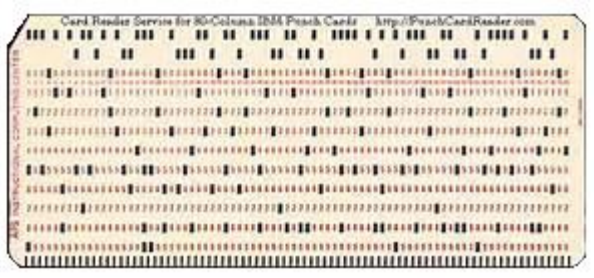
So how did you produce this code and how did you get it into the machine?

Pen & Paper

```
1000
1000
1000 78
1001 B1
1002 C8
1003 1A
1004 77
1005 13
1006 23
1007 0B
1008 C3 00 10
100B
```

```
; memcpy --
; Copy a block of memory from one location to another.
;
; Entry registers
;     BC - Number of bytes to copy
;     DE - Address of source data block
;     HL - Address of target data block
;
; Return registers
;     BC - Zero

org    1000h      ;Origin at 1000h
memcpy     public
loop   ld     a,b        ;Test BC,
       or     c         ;If BC = 0,
       ret    z         ;Return
       ld     a,(de)     ;Load A from (DE)
       ld     (hl),a     ;Store A into (HL)
       inc    de         ;Increment DE
       inc    hl         ;Increment HL
       dec    bc         ;Decrement BC
       jp     loop       ;Repeat the loop
       end
```



What was the next development?

Problem

Assembly Code difficult to learn and write for non-tech people, far too easy to make mistakes

Solution

A language more like ‘real’ language

The first such language was FORTRAN.

Born is September 1957 (60 years ago)

Example of FORTRAN code

```
program average

  ! Read in some numbers and take the average
  ! As written, if there are no data points, an average of zero is returned

  implicit none

  real, dimension(:), allocatable :: points
  integer                         :: number_of_points
  real                            :: average_points=0., positive_average=0., negative_average=0.

  write (*,*) "Input number of points to average:"
  read  (*,*) number_of_points

  allocate (points(number_of_points))

  write (*,*) "Enter the points to average:"
  read  (*,*) points

  ! Take the average by summing points and dividing by number_of_points
  if (number_of_points > 0) average_points = sum(points) / number_of_points

  ! Now form average over positive and negative points only
  if (count(points > 0.) > 0) then
    positive_average = sum(points, points > 0.) / count(points > 0.)
  end if

  if (count(points < 0.) > 0) then
    negative_average = sum(points, points < 0.) / count(points < 0.)
  end if

  deallocate (points)

  ! Print result to terminal
  write (*,'(a,g12.4)') 'Average = ', average_points
  write (*,'(a,g12.4)') 'Average of positive points = ', positive_average
  write (*,'(a,g12.4)') 'Average of negative points = ', negative_average

end program average
```

What was wrong?

Problem

While FORTRAN was much easier to use than assembler. It is still rather obtuse with special jargon. It was also designed for use in the scientific academic world.

Solution 1

A more general purpose language

The first such language was ALGOL.
The first generally used version being ALGOL 60

Example of ALGOL 60 code

```
begin
 integer N;
 Read Int(N);

 begin
  real array Data[1:N];
  real sum, avg;
  integer i;
  sum:=0;

  for i:=1 step 1 until N do
   begin real val;
    Read Real(val);
    Data[i]:=if val<0 then -val else val
   end;

  for i:=1 step 1 until N do
   sum:=sum + Data[i];
  avg:=sum/N;
  Print Real(avg)
 end
end
```

Almost readable by someone not trained in ALGOL!

Solution 2

A language designed for use in the business world

The language was COBAL.
Based on the ALGOL it is much more verbose language.
Designed initially to produce reports.

```
OPEN INPUT sales, OUTPUT report-out
    INITIATE sales-report

    PERFORM UNTIL 1 <> 1
      READ sales
        AT END
          EXIT PERFORM
      END-READ

      VALIDATE sales-record
      IF valid-record
        GENERATE sales-on-day
      ELSE
        GENERATE invalid-sales
      END-IF
    END-PERFORM

    TERMINATE sales-report
CLOSE sales, report-out
    .
```

What was the problem with these programs?

Although much simpler than Assembler Code they were still quite difficult to learn as they contained special syntax and control features.

Thus BASIC was born

Again its structure is based on ALGOL, but much simplified. It was designed as a language for teaching programming.

```
5 REM Hangman
10 REM set up screen
20 INK 0: PAPER 7: CLS
30 Let x=240; GO SUB 1000; REM DRAW man
40 PLOT 238,120: DRAW 4,0: REM mouth
100 REM set up word
110 INPUT w$: REM word to guess
120 let b=LEN w$: LET v$=" "
130 FOR n=2 to b: LET v$=v$+" "
140 NEXT n: REM v$=word guessed so far
150 LET c=0: LET d=0: REM guess & mistake counts
160 FOR n=0 TO b-1
170 PRINT AT 20,n;" ";
180 NEXT n: REM write 's instead of letters
200 INPUT "Guess a letter ";g$

290 IF v$=w$ THEN GOTO 500: REM word guessed
300 If v$<>u$ THEN GOTO 200: REM guess was a correct letter
400 REM draw next part of gallows
410 If d=8 THE GOTO 600: REM man hanged

500 REM free man
510 OVER 1: REM rub out man
520 LET x=240: GO SUB 1000

570 GO TO 800

600 REM hangman

850 RESTORE: GOTO 5

1000 REM draw amn at position x

1100 RETURN

2000 DATA 120, 68 .....
```

What was the problem with BASIC?

Like the other languages it was a sequentially structured language.

Lines of code were run in sequence (apart from internal loops).
With only the GO TO statement to enable a change of direction.

The GO SUB instruction had been added but this only removed the return GO TO instruction, replacing it with a RETURN statement.

Solution

To develop a structured language using reusable Procedures.

Named **Procedures** had there ‘address’ saved separately. All the programmer needed to do was write the Procedure and then call it from the main program sequence. Variable values could be sent to and returned from the procedure.

Most languages developed along these lines.

An example

PASCAL

Pascal is a procedural programming language, published in 1970, as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. Based on ALGOL 60 and initially used to teach students.

Used today because it is easy to compile to different platforms.

```
program Printing;

var i : integer;

procedure Print(j : integer);
begin
  ...
end;

begin { main program }
  ...
  Print(i);
end.
```

Other Developments during the 70's

LISP

The name LISP derives from "LISt Processor". Linked lists are one of Lisp's major data structures, and Lisp source code is made of lists.
First released a year after FORTRAN.

```
(defparameter *small* 1)
(defparameter *big* 100)

(defun guess-my-number ()
    (ash (+ *small* *big*) -1))

(defun smaller ()
    (setf *big* (1- (guess-my-number)))
    (guess-my-number))

(defun bigger ()
    (setf *small* (1+ (guess-my-number)))
    (guess-my-number))

(defun start-over ()
  (defparameter *small* 1)
  (defparameter *big* 100)
  (guess-my-number))
```

Other Developments during the 70's

ADA

ADA (named after Ada Lovelace) is a structured, statically typed, imperative, wide-spectrum, and object-oriented high-level computer programming language, extended from Pascal and other languages.

ADA was originally designed by a team under contract to the United States Department of Defense (DoD) from 1977 to 1983 to supersede over 450 programming languages used by the DoD at that time.

ADA was originally targeted at embedded and real-time systems, and is used, for example, by air-traffic control systems

```
task body Airplane is
   Rwy : Runway_Access;
 begin
   Controller1.Request_Takeoff (ID, Rwy); -- This call blocks until Controller task accepts and completes the accept block
   Put_Line (Airplane_ID'Image (ID) & "  taking off...");
   delay 2.0;
   Rwy.Cleared_Runway (ID);            -- call will not block as "Clear" in Rwy is now false and no other tasks should be inside protected object
   delay 5.0; -- fly around a bit...
   loop
     select   -- try to request a runway
       Controller1.Request_Approach (ID, Rwy); -- this is a blocking call - will run on controller reaching accept block and return on completion
       exit; -- if call returned we're clear for landing - leave select block and proceed...
     or
       delay 3.0;  -- timeout - if no answer in 3 seconds, do something else (everything in following block)
       Put_Line (Airplane_ID'Image (ID) & "   in holding pattern");  -- simply print a message
     end select;
   end loop;
   delay 4.0;  -- do landing approach...
   Put_Line (Airplane_ID'Image (ID) & "          touched down!");
   Rwy.Cleared_Runway (ID);  -- notify runway that we're done here.
 end;
```

What was the next step?

During the late 60’ and 70’s these programs were extended and refined, terminals with TV monitors were introduced and magnetic tape and discs meant that programs could be stored ready for use, they still need compiling before running but the reading from cards and storing the input before compiling was no longer necessary.

All these languages depended on compilers and interpreters. Interpreters take the code and convert it to assembly code and then the program runs. Compilers interpreted the code into machine code and then store the result in a program file (executable), This can then be run repeatably on the computer.
Compiled programs are quicker.
A machine specific assembly code is required therefore there are machine specific compilers.

C was developed as a way of simplifying compilation.

Why C?

C was designed to map much more efficiently to typical machine instructions it has been widely used in situations which formerly required assembly language such as operating systems and embedded systems.

An extension to accommodate object orientated programming was developed in C++ and Microsoft later added C# in the early 2000's as a general purpose object orientated language for their .NET environment.

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```
#include <iostream>

int main()
{
                std::cout << "Hello, world!\n";
}
```

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```

Still problems?

It was recognised in the late 80's that there was a problem with the machine dependencies causing implementation problems for programmers.

Thus was born JAVA,

designed as a "general-purpose computer programming language that is class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible".

JAVA enables a programmer to 'write once – run anywhere'.

Following compilation a JAVA program can run on any platform that supports JAVA without any need for re-compilation.

```
public class Puppy {
  public Puppy(String name) {
    // This constructor has one parameter, name.
    System.out.println("Passed Name is :" + name );
  }

  public static void main(String []args) {
    // Following statement would create an object myPuppy
    Puppy myPuppy = new Puppy( "tommy" );
  }
}
```

Does anyone find JAVA easy to read?

Not many, so **PYTHON** was invented

Python is an interpreted language, first released in 1991. Its design philosophy emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code.

Python has a large library of extensions which can be used to add functionality.

It is the principle programing tool for the Raspberry Pi and other LINUX systems.

```
parser = argparse.ArgumentParserfl
parser.add_argument("-m", default = False, action=,'store
true",  help="Counting Characters",  required=False)
parser.add_arqnrmen("J", default = False,  action=',store
tue",  help="Counting Lines",  required=False)
parser.add_argmment("-w", default = False, action=,,store
tue",  help="Counting Words",  required=False)
parser.add_argument('fllenames',  default = None,
heip="Fiienames ",  narys='"')

args = parser.parse_argsS
if args.filenames : None:
                print('No filenames given!')
else:
                for f in args.fllernmes:
                                pdn(O

if args.m: Ttue:
                print('-m is on!')
else:
                print('-m if offl')

if args.l: Ttue:
                print('-l is on!')
else:
                print('-l if off!')

if args.w: TYue:
                print('-w is on!')
else:
                print('-w if off!')
```

I don’t it find so easy to understand, but easier than Java

Any advance on Pen and paper!

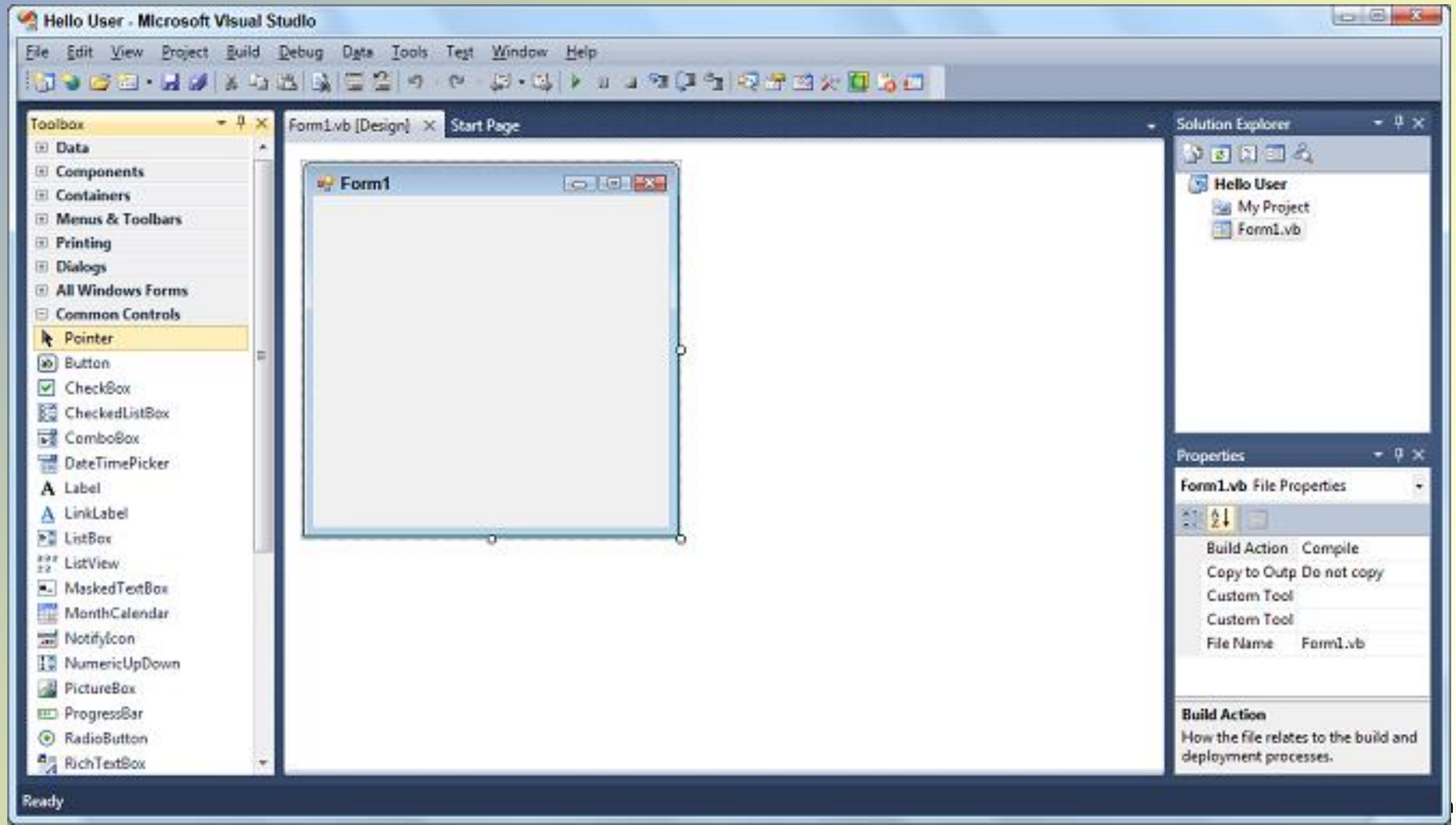
IDEs (Integrated Development Environment)

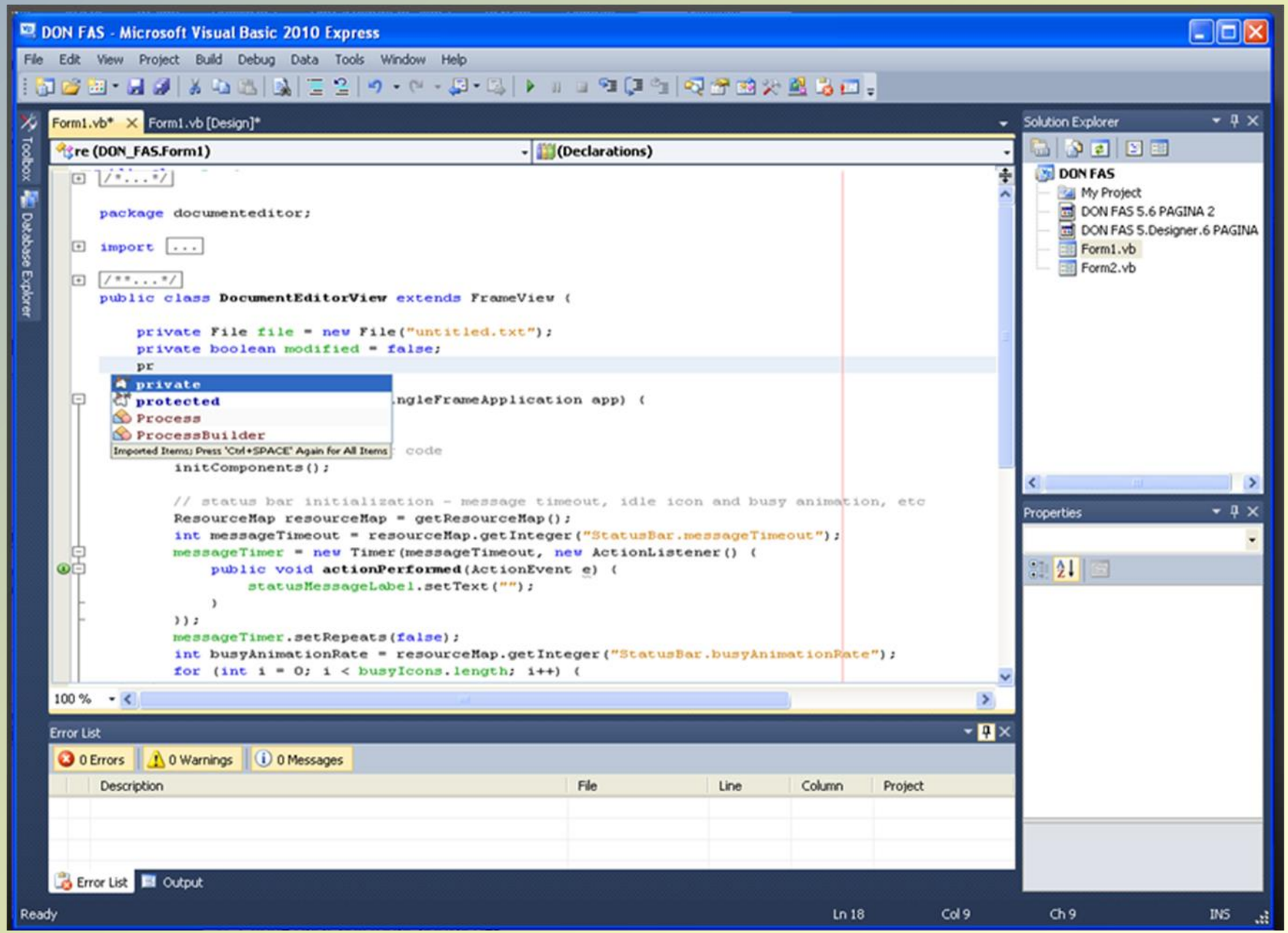
A piece of software which allows a programmer to:

- Design the layout
- Build using components
- Add subsidiary code in a code editor which has a syntax checker
- Run the code without compiling
- Get feedback on errors – syntax and logical
- To step through a section of code in debug mode
- Compile the code into an executable program

Etc.

The following example is from Visual Basic ca 2010, but most languages have something similar


Hello User - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Test Window Help
Toolbox
Data
Components
Containers
Menus & Toolbars
Printing
Dialogs
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
Form1.vb [Design]
Start Page
Form1
Solution Explorer
Hello User
My Project
Form1.vb
Properties
Form1.vb File Properties
Build Action Compile
Copy to Outp Do not copy
Custom Tool
Custom Tool
File Name Form1.vb
Build Action
How the file relates to the build and deployment processes.
Ready


DON FAS - Microsoft Visual Basic 2010 Express
File Edit View Project Build Debug Data Tools Window Help
Form1.vb* Form1.vb [Design]*
re (DON_FAS.Form1)
(Declarations)
/*...*/
package documenteditor;
import ...
/**...*/
public class DocumentEditorView extends FrameView {
private File file = new File("untitled.txt");
private boolean modified = false;
pr
private
protected
Process
ProcessBuilder
Imported Items: Press 'Ctrl+SPACE' Again for All Items
ngleFrameApplication app) {
code
initComponents();
// status bar initialization - message timeout, idle icon and busy animation, etc
ResourceMap resourceMap = getResourceMap();
int messageTimeout = resourceMap.getInteger("StatusBar.messageTimeout");
messageTimer = new Timer(messageTimeout, new ActionListener() {
public void actionPerformed(ActionEvent e) {
statusMessageLabel.setText("");
}
});
messageTimer.setRepeats(false);
int busyAnimationRate = resourceMap.getInteger("StatusBar.busyAnimationRate");
for (int i = 0; i < busyIcons.length; i++) {
100 %
Solution Explorer
DON FAS
My Project
DON FAS 5.6 PAGINA 2
DON FAS 5.Designer.6 PAGINA
Form1.vb
Form2.vb
Properties
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Error List
Output
Ready
Ln 18
Col 9
Ch 9
INS

Developments in database programs

During the 80's powerful databases were developed and methods of integrating these into applications were needed.

A leading company here was Oracle and their SQLForms.

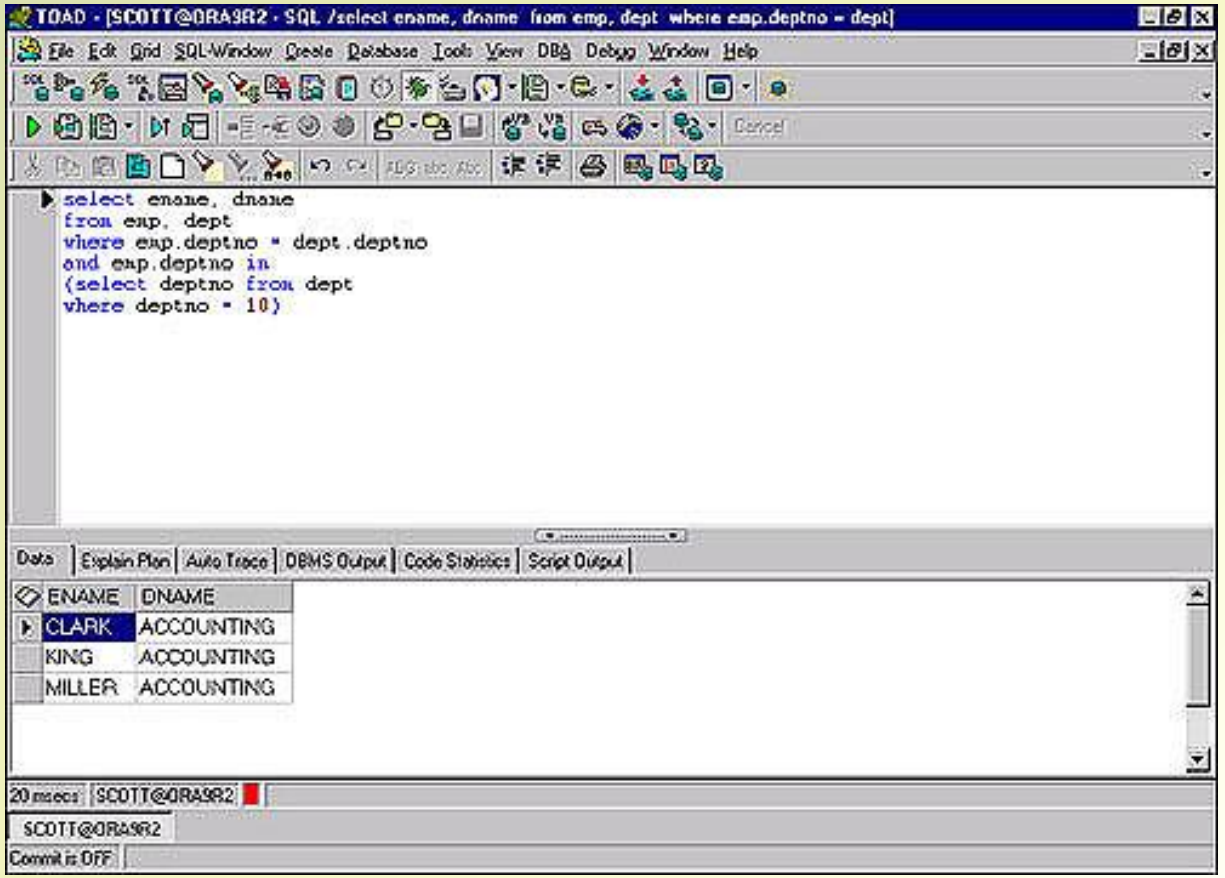
Using SQL Forms in an integrated environment a programmer could produce applications for inputting and viewing database information and produce reports which could be printed. Each screen page was a 'Form'.

PL/SQL was a procedural language by which a programmer could manipulate the data before it was displayed or printed.

These procedures were saved in the database, and could be accessed by other Forms.

The TOAD code editor is a useful tool for an SQL programmer, allowing access to the database and a debugging environment

```
1> CREATE OR REPLACE PROCEDURE employer_details
2> IS
3>  CURSOR emp_cur IS
4>  SELECT first_name, last_name, salary FROM emp_tbl;
5>  emp_rec emp_cur%rowtype;
6> BEGIN
7>  FOR emp_rec in sales_cur
8>  LOOP
9>  dbms_output.put_line(emp_cur.first_name || ' ' ||emp_cur.last_name
10>    || ' ' ||emp_cur.salary);
11> END LOOP;
12>END;
13> /
```



The Influence of the Web on programming

From an early stage in the mid-90's it was recognised that a programming language enabling the integration of databases with web pages was required.

Began with simple input and output routines saved in the cgi directory on the web server.
Written in notepad. EG. Microsoft had a method of accessing Access databases.

This was quickly followed by the development of more powerful 'Scripting Languages' which could be used not only to access databases but to build the actual web pages interactively.

Examples of the these languages are
PERL
PHP
JAVAscript
Vbscript
These last 2 can also be used on client machines to interactively re-draw a web page

PERL

Perl was originally developed as a general-purpose Unix scripting language to make report processing easier.
PERL gained popularity in the 1990's as a CGI scripting language.
Popular on LINUX based systems

```
#!/usr/bin/perl
use strict;
use warnings;
use IO::Handle;

my ( $remaining, $total );

$remaining = $total = shift(@ARGV);

STDOUT->autoflush(1);

while ( $remaining ) {
   printf ( "Remaining %s/%s \r", $remaining--, $total );
   sleep 1;
}

print "\n";
```

PHP

PHP is a server-side scripting language designed primarily for web development but also used as a general-purpose programming language. Originally created in 1994.

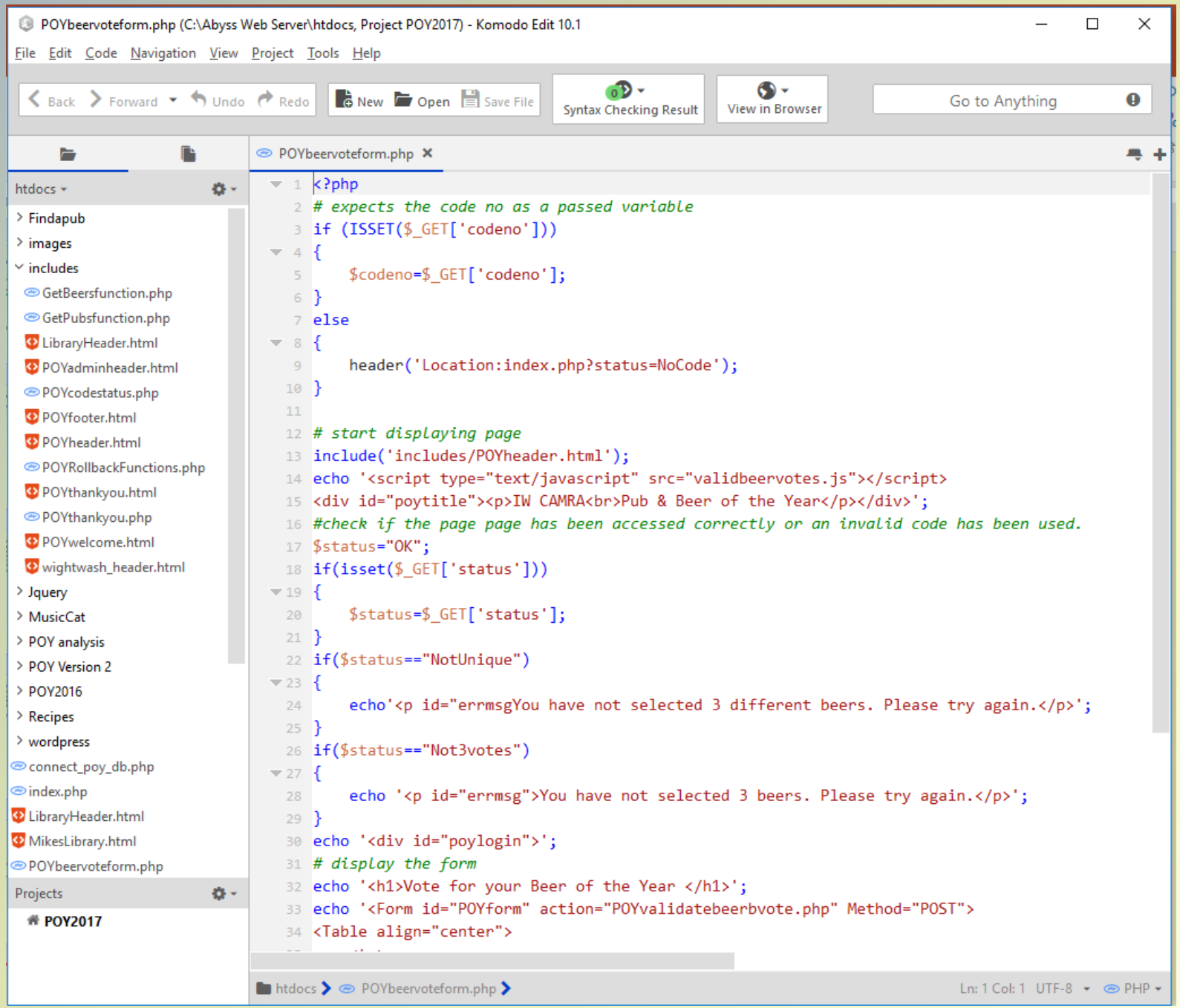
PHP code may be embedded into HTML or HTML5 code, or it can be used in combination with various web template systems, web content management systems and web frameworks.

PHP code is processed by a PHP interpreter on the web server The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page.

The following example is taken from a recipes app I have written so that I can access my large stock of recipes from my tablet while in the kitchen.

Process input describing a new ingredient and stores this data in the recipes database.

```
<?php
   require('connect_nut_db.php');
if(isset($_POST['unititem']))
{
   $item=$_POST['unititem'];
   $from=$_POST['unitfrom'];
   $to=$_POST['unitto'];
   $qty=$_POST['amountto'];
   if($item=="")
   {
      $query="INSERT INTO tbl_convert (descr, to_unit, to_amount) VALUES ('$from', '$to', '$qty')";
   }
   else
   {
      $query="INSERT INTO tbl_convert (item, descr, to_unit, to_amount) VALUES ('$item', '$from', '$to', '$qty')";
   }
   $r=mysqli_query($dbc, $query);
   if (mysqli_affected_rows($dbc)!="1")
   {
      echo 'no row added. '.mysqli_error($dbc);
   }
   mysqli_close($dbc);
}
else
{
   echo ' no values posted';
   exit();
}
header('Location:Conversions.php');
?>
```


POYbeervoteform.php (C:\Abyss Web Server\htdocs, Project POY2017) - Komodo Edit 10.1
File Edit Code Navigation View Project Tools Help
Back Forward Undo Redo New Open Save File Syntax Checking Result View in Browser Go to Anything
htdocs
Findapub
images
includes
GetBeersfunction.php
GetPubsfunction.php
LibraryHeader.html
POYadminheader.html
POYcodestatus.php
POYfooter.html
POYheader.html
POYRollbackFunctions.php
POYthankyou.html
POYthankyou.php
POYwelcome.html
wightwash_header.html
Jquery
MusicCat
POY analysis
POY Version 2
POY2016
Recipes
wordpress
connect_poy_db.php
index.php
LibraryHeader.html
MikesLibrary.html
POYbeervoteform.php
Projects
POY2017
POYbeervoteform.php
<?php
# expects the code no as a passed variable
if (ISSET($_GET['codeno']))
{
$codeno=$_GET['codeno'];
}
else
{
header('Location:index.php?status=NoCode');
}
# start displaying page
include('includes/POYheader.html');
echo '<script type="text/javascript" src="validbeervotes.js"></script>
IW CAMRA
Pub & Beer of the Year ';
#check if the page page has been accessed correctly or an invalid code has been used.
$status="OK";
if(isset($_GET['status']))
{
$status=$_GET['status'];
}
if($status=="NotUnique")
{
echo'<p id="errmsgYou have not selected 3 different beers. Please try again. ';
}
if($status=="Not3votes")
{
echo ' You have not selected 3 beers. Please try again. ';
}
echo ' ';
display the form
echo '<h1>Vote for your Beer of the Year </h1>';
echo '<Form id="POYform" action="POYvalidatebeerbvote.php" Method="POST">
<Table align="center">
htdocs POYbeervoteform.php
Ln: 1 Col: 1 UTF-8 PHP

That’s all Folks