Continue

We've updated our Privacy Policy effective December 15. Please read our updated Privacy Policy and tap The following is a list of Hello, world! programs.Hello, world! programs make the text "Hello, world!" appear on a computer screen. It is usually the first program encountered when learning a programming language. Otherwise, it's a basic sanity check for an installation of a new programming language. If "Hello World" does not run, one must not try to develop complex programs before fixing the issues with the installation.For even more languages have a look at the Hello World Collection. It should be noted that the 4DOS/4NT batch language is a superset of the MS-DOS batch language. message "Hello, world!" with style = popup; REPORT ZHELLO. START-OF-SELECTION. WRITE "Hello, world!". The example below makes use of the singleton pattern and outputs the text in a message box instead of a classic list output. REPORT ZHELLO. CLASS lcl_hello DEFINITION CREATE PRIVATE FINAL. PUBLIC SECTION. CLASS-DATA self TYPE REF TO lcl_hello READ-ONLY. CLASS-METHODS class_constructor. METHODS say_hello. PRIVATE SECTION. CONSTANTS con_hello_world TYPE c LENGTH 13 VALUE 'Hello, World!'. ENDCLASS. CLASS lcl_hello IMPLEMENTATION. METHOD class_constructor. CREATE OBJECT lcl_hello=>self. ENDMETHOD. METHOD say_hello. MESSAGE con_hello_world TYPE 'I'. ENDMETHOD. ENDCLASS. START-OF-SELECTION. lcl_hello=>self->say_hello( ). WRITE "Hello, world!" This will output to the output window only, which an end user would not see. This version will be visible to the end user.var helloWorld:TextField = this.createTextField("helloWorld", this.getNextHighestDepth(), 1, 1, 100, 20);helloWorld.text = "Hello, world!"; package{ public class HelloWorld { public function HelloWorld() { trace("Hello, world!"); } }} with Ada.Text_IO; procedure Hello isbegin Ada.Text_IO.Put_Line ("Hello, world!");end Hello; The ALGOL 68 standard requires that reserved-words, types and operators are in a different typeface.Hence programs are typically published in either bold or an underline typeface, e.g.:begin printf($"Hello, world!"l$)endIn the popular upper-case stropping convention for bold words:BEGIN printf($"Hello, world!"l$)ENDor using a wikitext like quote stropping, this is especially suitable on computers with only 6 bits per character (hence only have UPPERCASE):'BEGIN' PRINTF($"HELLO, WORLD!"L$)'END'or minimally using the "brief symbol" form of begin and end.(printf($"Hello, world!"l$)) PROC main() WriteF('Hello, world!');ENDPROC This program sends the message out via the Diagnostics Interface after start-up.program_name = 'Hello'define_startsend_string 0,'Hello World!' RHWPGM [1] R'HELLO WORLD!' The Del on the first line begins function definition for the program named HWPGM. It is a niladic function (no parameters, as opposed to monadic or dyadic) and it will return an explicit result which allows other functions or APL primitives to use the returned value as input.The line labeled 1 assigns the text vector 'Hello, world!!' to the variable RThe last line is another Del which ends the function definition.When the function is executed by typing its name the APL interpreter assigns the text vector to the variable R, but since we have not used this value in another function, primitive, or assignment statement the interpreter returns it to the terminal, thus displaying the words on the next line below the function invocation.The session would look like this HWPGMHello, world!!While not a program, if you simply supplied the text vector to the interpreter but did not assign it to a variable it would return it to the terminal as output. Note that user input is automatically indented 6 spaces by the interpreter while results are displayed at the beginning of a new line. 'Hello, world!'Hello, world!! or:display dialog "Hello, world!" or simply: // in the page behind using C#protected void Page_Load(object sender, EventArgs e){ Response.Write("Hello, world!");}' in the page behind using VB.NETProtected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs) Response.Write("Hello, world!")End Sub// ASPX Page Template or orHello World! See the example program in the Wikipedia PDP-8 article. bdos equ 0005H ; BDOS entry point start: mvi c,9 ; BDOS function: output string lxi d,msg$; address of msg call bdos ret ; return to CCP msg$: db 'Hello, world!$' end start 10 ORG #8000 ; Start address of the routine 20 START LD A,2 ; set the output channel 30 CALL #1601 ; to channel 2 (main part of TV display) 40 LD HL,MSG ; Set HL register pair to address of the message 50 LOOP LD A,(HL) ; De-reference HL and store in A 60 AND A ; Null terminator? 70 RET Z ; If so, return 80 RST #10 ; Print the character in A 90 INC HL ; HL points at the next char to be printed 100 JR LOOP 110 RET 120 MSG DEFM "Hello, world!" 130 DEFB 13 ; carriage return 140 DEFB 0 ; null terminator A_CR = $0D ;carriage return BSOUT = $FFD2 ;kernel ROM sub, write to current output device ; LDX #$00 ;starting index in .X register ; LOOP LDA MSG,X ;read message text BEQ LOOPEND ;end of text ; JSR BSOUT ;output char INX BNE LOOP ;repeat ; LOOPEND RTS ;return from subroutine ; MSG .BYT 'Hello, world!',A_CR,$00 See the example section of the Nova article. MODEL SMALLIDEALSTACK 100HDATASEG MSG DB 'Hello, world!', 13, '$'CODESEGStart: MOV AX, @data MOV DS, AX MOV DX, OFFSET MSG MOV AH, 09H ; DOS: output ASCII$ INT 21H MOV AX, 4C00H INT 21HEND Start .MODEL Small.STACK 100h.DATA db msg 'Hello, world!$'.CODEstart: mov ah, 09h lea dx, msg ; or mov dx, offset msg int 21h mov ax,4C00h int 21hend start ; FASM example of writing 16-bit DOS .COM program; Compile: "FASM HELLO.ASM HELLO.COM" org $100 use16 mov ah,9 mov dx,xhello int $21 ; DOS call: text output mov ah,$4C int $21 ; Return to DOSxhello db 'Hello world!!!$' Example of making 32-bit PE program as raw code and data:format PE GUI entry startsection '.code' code readable executable start: push 0 push _caption push _message push 0 call [MessageBox] push 0 call [ExitProcess]section '.data' data readable writeable _caption db 'Win32 assembly program',0 _message db 'Hello, world!',0section '.idata' import data readable writeable dd 0,0,0,RVA kernel_name,RVA kernel_table dd 0,0,0,RVA user_name,RVA user_table dd 0,0,0,0,0 kernel_table: ExitProcess dd RVA _ExitProcess dd 0 user_table: MessageBox dd RVA _MessageBoxA dd 0 kernel_name db 'KERNEL32.DLL',0 user_name db 'USER32.DLL',0 _ExitProcess dw 0 db 'ExitProcess',0 _MessageBoxA dw 0 db 'MessageBoxA',0section '.reloc' fixups data readable discardableUsing FASM import macro, unicode (MessageBoxW is one of few unicode functions 'supported' by Windows 9x/ME) and section sharing, no relocation (not required for 32-bit Windows NT executables, recommended for DOS-based Windows, required for x64), no heap - Not a beginners example but only 1024 instead of 3072 bytes:format PE GUI 4.0heap 0entry startinclude 'win32a.inc' section '.text' code import readable executable data library kernel, 'KERNEL32.DLL',\ user,'USER32.DLL' import kernel,\ ExitProcess, 'ExitProcess' import user,\ MessageBoxW, 'MessageBoxW' start: xor ebx, ebx push ebx push ebx push _message push ebx call [MessageBoxW] push ebx call [ExitProcess] _message du 'Hello, world!' ,0section '.reloc' fixups data readable discardable format ELF executableentry _start_start: mov eax, 4 mov ebx, 1 mov ecx, msg mov edx, msg_len int 80h mov ebx, 0 mov eax, 1 int 80h msg db 'Hello, world!', 0xA msg_len = $-msg .datamsg: .ascii "Hello, world!" len = . - msg.text .global _start_start: movl $len,%edx movl $msg,%ecx movl $1,%ebx movl $4,%eax int $0x80 movl $0,%ebx movl $1,%eax int $0x80 section .datamsg db 'Hello, world!',0xAlen equ $-msg section .textglobal _start_start: mov edx,len mov ecx,msg mov ebx,1 mov eax,4 int 0x80 mov ebx,0 mov eax,1 int 0x80 extern printf ; Request symbol "printf".global main ; Declare symbol "main".section .data str: DB "Hello World!", 0x0A, 0x00section .textmain: PUSH str ; Push string pointer onto stack. CALL printf ; Call printf. POP eax ; Remove value from stack. MOV eax,0x0 ; \ Return value 0. RET ; / TERM EQU 19 console device no. (19 = typewriter) ORIG 1000 start addressSTART OUT MSG(TERM) output data at address MSG HLT halt executionMSG ALF "HELLO" ALF " WORL" ALF "D " END START end of program string BYTE "Hello, world!",#a,0 string to be printed (#a is newline and 0 terminates the string) Main GETA $255,string get the address of the string in register 255 TRAP 0,Fputs,StdOut put the string pointed to by register 255 to file StdOut TRAP 0,Halt,0 end process .MCALL .REGDEF,.TTYOUT,.EXIT .EXIT REGDEF HELLO: MOV #MSG,R1 MOVB (R1)+,R0 BEQ EXIT LOOP: .TTYOUT BR LOOPEXIT: .EXIT MSG: .ASCIZ /Hello, world!/ .END HELLO MOV #TXT,R1 ;Moving string address to R1 CLR R2 ;String length=0, means null will be the termination character EMT 20 ;Print the string HALT TXT: .ASCIZ /Hello, world!/ .END include lvo/exec_lib.i include lvo/dos_lib.i ; open DOS library movea.l 4.w,a6 lea dosname(pc),a1 moveq #36,d0 jsr _LVOOpenLibrary(a6) movea.l d0,a6 ; actual print string lea hellostr(pc),a0 move.l a0,d1 jsr _LVOPutStr(a6) ; close DOS library movea.l a6,a1 movea.l 4.w,a6 jmp _LVOCloseLibrary(a6)dosname dc.b 'dos.library',0hellostr dc.b 'Hello, world!',0 ;print move.l #Hello,-(A7) move.w #9,-(A7) trap #1 addq.l #6,A7;wait for key move.w #1,-(A7) trap #1 addq.l #2,A7;exit clr.w -(A7) trap #1Hello dc.b 'Hello, world!',0 pea (strign) ; push string address onto stack dc.w $FF09 ; call DOS "print" by triggering an exception addq.l #4,a7 ; restore the stack pointer dc.w $FF00 ; call DOS "exit" strign: dc.b "Hello, world!",13,10,0 .title hello .psect data, wrt, noexe chan: .blkw 1 iosb: .blkq 1 term: .ascid "SYS$OUTPUT" msg: .ascii "Hello, world!" len = . - msg .psect code, nowrt, exe .entry hello, ^m ; Establish a channel for terminal I/O $assign_s devnam=term, - chan=chan blbc r0, end ; Queue the I/O request $qiow_s chan=chan, - func=#io$_writevblk, - iosb=iosb, - p1=msg, - p2=#len ; Check the status and the IOSB status blbc r0, end movzwl iosb, r0 ; Return to operating system end: ret .end hello HELLO CSECT The name of this program is 'HELLO' USING *,12 Tell assembler what register we are using SAVE (14,12) Save registers LR 12,15 Use Register 12 for this program WTO 'Hello, world!' Write To Operator RETURN (14,12) Return to calling party END HELLO This is the end of the program .program ADR R0, message SWI "OS_Write0" SWI "OS_Exit" .message EQUS "Hello, world!" EQUB 0 ALIGNor the even smaller version (from qUE); SWI "OS_WriteS":EQUS "Hello, world!":EQUB0:ALIGN:MOV PC,R14 .data msg: .asciiz "Hello, world!" .align 2 .text .globl main main: la $a0,msg li $v0,4 syscall jr $ra .data msg: .ascii "Hello, world!" len = . - msg .text .globl _main _main: li r0, 4 ; write li r3, 1 ; stdout addis r4, 0, ha16(msg) ; high 16 bits of address addi r4, r4, lo16(msg) ; low 16 bits of address li r5, len ; length sc li r0, 1 ; exit li r3, 0 ; exit status sc SYSTEM BPM START M:PRINT (MESS,HW) M:EXIT HW TEXTC 'HELLO WORLD' END START (The comma after the command name is optional.) MsgBox(0,'','Hello, world!') MsgBox("Hello, world!","aTitle") BEGIN { print "Hello, world!" } This is the first known Hello, world! program ever written:[1] main() { extrn a, b, c; putchar(a); putchar(b); putchar(c); putchar('!*n'); } a 'hell'; b 'o, w'; c 'orld'; Also known as Triton Tools on older versions. On Baan ERP you can create a program on 3GL or 4GL mode. function main(){ message("Hello, world!")} choice.cont.process:on.choice: message("Hello, world!")On this last case you should press the Continue button to show the message. or or using the C preprocessor#!/bin/bash#define cpp #cpp $0 2> /dev/null | /bin/bash; exit $?#undef cpp#define HELLO_WORLD echo "hello, world"HELLO_WORLD | tr a-z A-Z The following example works for any ANSI/ISO-compliant BASIC implementation, as well as most implementations built into or distributed with microcomputers in the 1970s and 1980s (usually some variant of Microsoft BASIC):10 PRINT "Hello, world!"20 ENDNote that the "END" statement is optional in many implementations of BASIC.Some implementations could also execute instructions in an immediate mode when line numbers are omitted. The following examples work without requiring a RUN instruction. Later implementations of BASIC allowed greater support for structured programming and did not require line numbers for source code. The following example works when RUN for the vast majority of modern BASICs.PRINT "Hello, world!"ENDAgain, the "END" statement is optional in many BASICs. Print "Hello, world!"WaitKey orTEXT 0,0,"Hello, world!"WAIT KEYNote: In the "classic" Dark Basic the WAIT KEY command is optional as the console goes up when the program has finished. PRINT "Hello World"SLEEPENDor or'without a newline? "Hello World"; AddText "Hello, world!"DrawScreenWaitKey 10 PRINT "Hello, World!"20 END To write to the main window: Or drawn in a graphics window:nomainwinopen "Hello, world!" for graphics as #mainprint #main, "place 50 50"print #main, "\Hello, world!"print #main, "flush"wait TextWindow.WriteLine("Hello, world!") DEBUG "Hello, world!", CRor, the typical microcontroller Hello, world! program equivalent with the only output device present being a light-emitting diode (LED) (in this case attached to the seventh output pin): DO HIGH 7 'Make the 7th pin go high (turn the LED on) PAUSE 500 'Sleep for half a second LOW 7 ' Make the 7th pin go low (turn the LED off) PAUSE 500 'Sleep for half a second LOOP END sub main print "Hello, world!" end sub OpenConsole() PrintN("Hello, world!") Input()orMessageRequester("Hello, World","Hello, World")or PRINT "Hello, World" ENDSame for QBasic On TI calculators of the TI-80 through TI-86 range::Disp "Hello, world! (note the optional ending quotes):Output(X,Y,"Hello, world! (note the optional ending parenthesis):Text(X,Y,"Hello, world! (writes to the graph rather than home screen):Text(-1,X,Y,"Hello, world! (only on the 83+ and higher, provides larger text, home screen size):"Hello, world! (last line of program only)Note: "!" character is not on the keypad. It can be accessed from "Catalog" menu, "Probability" menu, or on "Math" menu (as factorial notation). On TI-89/TI-89 Titanium/TI-92(+)/Voyage 200 calculators:.hellowld():Prgm:Disp "Hello, world!":EndPrgm Public Sub Main() Debug.Print "Hello, world!"End SuborPublic Sub Main() MsgBox "Hello, world!"End SuborPrivate Sub Form_Activate() Print "Hello, world!"End SubAlternatively, copy this into a New Form:Private Sub Form_Click() Form1.Hide Dim HelloWorld As New Form1 HelloWorld.Width = 2500: HelloWorld.Height = 1000: HelloWorld.Caption = "Hello, world!": HelloWorld.CurrentX = 500: HelloWorld.CurrentY = 75 HelloWorld.Show: HelloWorld.Font = "Tahoma": HelloWorld.FontBold = True: HelloWorld.FontSize = 12: HelloWorld.Print "Hello, world!"End Sub Module HelloWorldApp Sub Main() System.Console.WriteLine("Hello, world!") End SubEnd Module In addition to the ANSI syntax at the head of this article, most Pick operating system flavors of Dartmouth BASIC support extended syntax allowing cursor placement and other terminfo type functions for VDT'sX, Y positioning (colon ":" is the concatenation instruction): PRINT @(34,12) : "Hello, world!"Will display the string "Hello, world!" roughly centered in a 80X24 CRT.Other functions: PRINT @(-1) : @(34,12) : "Hello, world!"Will clear the screen before displaying the string "Hello, world!" roughly centered in a 80X24 CRT.Syntax variants: CRT "Hello, world!"Supporting the "@" functions above, the CRT statement ignores previous PRINTER statements and always sends output to the screen.Some Pick operating system environments such as OpenQM support the DISPLAY variant of PRINT. This variant in addition to the "@" functions maintains pagination based upon the settings of the TERM variable: DISPLAY "Hello, world!" or@echo offset hellostring=Hello World!echo %hellostring%or@echo offecho Hello World!pauseexitor@echo Hello World!pauseexit @echo offecho Content-type: text/plain >> sample.cgiecho. >> sample.cgiecho. >> sample.cgiecho Hello, world! >> sample.cgi "Hello, world!"or, with the newline GET "LIBHDR"LET START () BE$(WRITES ("Hello, world!*N")$) T 1 "Hello, World" 0 J(1 TM 5 ZV 3 "Hello, world") %TITLE 'HELLO_WORLD'MODULE HELLO_WORLD (IDENT='V1.0', MAIN=HELLO_WORLD, ADDRESSING_MODE (EXTERNAL=GENERAL)) =BEGIN LIBRARY 'SYS$LIBRARY:STARLET'; EXTERNAL ROUTINE LIB$PUT_OUTPUT;GLOBAL ROUTINE HELLO_WORLD =BEGIN LIB$PUT_OUTPUT(%ASCID %STRING('Hello, world!'))END;ENDELUDOM SuperStrict Graphics 640, 480, 0, 60 Local running:Int = 1 While running Cls DrawText "Hello World!", 1, 1 Flip If GetChar() running = 0 EndIf Wend End See also GUI section. WRITE ELEMENT:Earth 210 230 40 CENTER TEXT "Hello World!" #include int main(){ printf("Hello, world!"); return 0;} #include int main(void) { printf("Hello, World!"); return 0;} Class Test.Hello Extends %CSP.Page [ProcedureBlock] { ClassMethod OnPage() As %Status { &html Write "Hello, world!",! &html Quit $$$OK } } This program will work on the Avasmath 80 online programmable calculator.#BTN A1#PRI "HELLO WORLD!"#END OBJECT Codeunit 50000 HelloWorld{ PROPERTIES { OnRun=BEGIN MESSAGE(Txt001); END; } CODE { VAR Txt001@1000000000: TextConst 'ENU=Hello, world!'; BEGIN { Hello, world! in C/AL (Microsoft Business Solutions-Navision) } END. }} This program will work on the fx-9750 graphing calculator and compatibles."Hello, world!"orLocate 1,1,"Hello, world!" call echo("Hello, world!") The above C code can run in Ch as examples. The simple one in Ch is: printf("Hello, world!"); ; namespace HelloWorld; interface type HelloClass = class public class method Main; end; implementation class method HelloClass.Main;begin System.Console.WriteLine('Hello, world!');end; end. .assembly Hello {}.assembly extern mscorlib {}.method static void Main(){ .entrypoint .maxstack 1 ldstr "Hello, world!" call void [mscorlib]System.Console::WriteLine(string) ret} System.dllHello, world!or Function | Main WriteLine | "Hello, world"End | MainThe '|' represents the separation of the two text fields in the Cube standard IDE. #include int main() { std::cout This creates a message box saying "Hello, world!".OutFile "HelloWorld.exe"Name "Hello, world!"Caption "Hello, world!" Section Hello, world!SectionEnd Function .onInit MessageBox MB_OK "Hello, world!" QuitFunctionEnd Uses lablgtk let () = let window = GWindow.window ~title:"Hello" ~border_width:10 () in window#connect#destroy ~callback:GMain.Main.quit; let button = GButton.button ~label:"Hello World" ~packing:window#add () in button#connect#clicked ~callback:window#destroy; window#show (); GMain.Main.main () (On Psion Series 3 and later compatible PDAs.)PROC guihello: ALERT("Hello, world!","","Exit")ENDPorPROC hello: dINIT "Window Title" dTEXT "","Hello, world!" dBUTTONS "OK",13 DIALOGENDP Patch as ASCII-art:[Hello, world!(|[print]Patch as sourcecode:#N canvas 0 0 300 300 10;#X msg 100 150 Hello, world!;#X obj 100 200 print;#X connect 0 0 1 0; from Tkinter import Tk, Label root = Tk()Label(root, text="Hello, world!").pack() root.mainloop()Using PyQt:import sysfrom PyQt4.QtCore import *from PyQt4.QtGui import * app = QApplication(sys.argv)label = QLabel("Hello, World!")label.show()sys.exit(app.exec_()) from gtk import * label = Label("Hello, world!")label.show() window = Window()window.add(label)window.show() main() import pygameimport sys pygame.init()screen = pygame.display.set_mode((512, 256), 0, 32)f = pygame.font.SysFont(None, 32)t = f.render("Hello, world!", True, (255, 255, 255))tR = t.get_rect()screen.blit(t, tR)while True: for event in pygame.event.get(): if event.type == pygame.QUIT: pygame.quit() sys.exit() pygame.display.update() Kivy multi-platform frameworkimport kivy from kivy.app import Appfrom kivy.uix.label import Label class MyApp(App): def build(self): return Label(text='Hello world') MyApp().run() #include #include int main(int argc, char * argv[]) { QApplication app(argc, argv); QMessageBox::information(0, "Qt4", "Hello World!"); }or #include #include #include #include class HelloWorld : public QWidget { Q_OBJECT public: HelloWorld(); virtual ~HelloWorld(); public slots: void handleButtonClicked(); QPushButton *mPushButton; }; HelloWorld::HelloWorld() : QWidget(), mPushButton(new QPushButton("Hello, world!", this)) { connect(mPushButton, SIGNAL(clicked()), this, SLOT(handleButtonClicked())); } HelloWorld::~HelloWorld() {} void HelloWorld::handleButtonClicked() { std::cout addWidget(hello); layout->setMargin(10); layout->setSpacing(10); window->show(); return app.exec(); } view layout [text "Hello, world!"] view [text "Hello, world!"] * "Hello, world!"end (On Hewlett-Packard HP-48G and HP-49G series calculators.)> Hello ()TEXT "Hello, world!" require 'wxruby' class HelloWorldApp < Wx::App def on_init ourFrame = Wx::Frame.new(nil, -1, "Hello, world!").show ourDialogBox = Wx::MessageDialog.new(ourFrame, "Hello, world!", "Information:", \ Wx::OK|Wx::ICON_INFORMATION).show_modal endend HelloWorldApp.new.main_loop require 'gtk2' Gtk.initwindow = Gtk::Window.newwindow.signal_connect("delete_event") { Gtk.main_quit; false }button = Gtk::Button.new("Hello, world!")button.signal_connect("clicked") { Gtk.main_quit; false }window.add(button)window.show_allGtk.main require 'tk' window = TkRoot.new { title 'Hello, world!' }button = TkButton.new(window) { text 'Hello, world!' command proc { exit } pack} Tk.mainloop Evaluate in a workspace:Dialog confirm: 'Hello, world!'Using the Morphic GUI toolkit of Squeak Smalltalk:('Hello, world!' asMorph openInWindow) submorphs second color: Color blackUsing wxSqueak:Wx messageBox: 'Hello, world!' import org.eclipse.swt.SWT;import org.eclipse.swt.layout.RowLayout;import org.eclipse.swt.widgets.Display;import org.eclipse.swt.widgets.Shell;import org.eclipse.swt.widgets.Label;public class SWTHello { public static void main (String [] args) { Display display = new Display (); final Shell shell = new Shell(display); RowLayout layout = new RowLayout(); layout.justify = true; layout.pack = true; shell.setLayout(layout); shell.setText("Hello, world!"); Label label = new Label(shell, SWT.CENTER); label.setText("Hello, world!"); shell.pack(); shell.open (); while (!shell.isDisposed ()) { if (!display.readAndDispatch ()) display.sleep (); } display.dispose (); }} label .l -text "Hello, world!"pack .land the same in one linepack [label .l -text "Hello, world!"] package require Tktk_messageBox -message "Hello, world!"orpackage require Tkpack [button .b -text "Hello, world!" -command exit] Ubercode 1 class Hello public function main() code call Msgbox("Hello", "Hello, world!") end function end class message "Hello, world!" void main () { String s = "Hello World."; bc.OutputToConsole (s); } Sub Main() MsgBox "Hello, world!"End Sub Private Sub Form_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load MessageBox.Show("Hello, world!") Me.Close() End SubNote that the previous example will only work when the code is entered as part of a Form Load Event, such as the one created by default when generating a new project in the Visual Studio programming environment. Equivalently, the following code is roughly equivalent to the traditional Visual Basic 6 code by disabling the Application Framework and setting 'Sub Main' as the entry point for the application:Public Module MyApplication Sub Main() MessageBox.Show("Hello, world!") End SubEnd Classor using a class;Public Class MyApplication Shared Sub Main() MessageBox.Show("Hello, world!") End SubEnd Class #include @"pfc\vpi\vpi.ph" goal vpiCommonDialogs::note("Hello, world!"). This uses the Windows API to create a full window containing the text./* Name: Win32 example Copyright: GLP Author: Ryon S. Hunter Date: 20/03/07 17:11 Description: This is an example of what a Win32 hello world looks like.*/ #include #define APPTITLE "Win32 - Hello world" BOOL InitInstance(HINSTANCE,int);ATOM MyRegisterClass(HINSTANCE);LRESULT CALLBACK WinProc(HWND,UINT,WPARAM,LPARAM); LRESULT CALLBACK WinProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam){ PAINTSTRUCT ps; COLORREF c = RGB(0, 0, 0); HDC hdc; RECT rt; switch(message) { case WM_DESTROY: // Exit the window? Ok PostQuitMessage(0); break; case WM_PAINT: GetClientRect(hWnd, &rt); hdc = BeginPaint(hWnd, &ps); DrawText(hdc, "Hello world!", sizeof("Hello world!"), &rt, DT_CENTER); EndPaint(hWnd, &ps); break; } return DefWindowProc(hWnd,message,wParam,lParam);} ATOM MyRegisterClass(HINSTANCE hInstance){ WNDCLASSEX wc; wc.cbSize = sizeof(WNDCLASSEX); wc.style = CS_HREDRAW | CS_VREDRAW; wc.lpfnWndProc = (WNDPROC)WinProc; wc.cbClsExtra = 0; wc.cbWndExtra = 0; wc.hInstance = 0; wc.hIcon = NULL; wc.hCursor = LoadCursor(NULL, IDC_ARROW); wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH); wc.lpszMenuName = NULL; wc.lpszClassName = APPTITLE; wc.hIconSm = NULL; return RegisterClassEx(&wc);} BOOL InitInstance(HINSTANCE hInstance, int nCmdShow){ HWND hWnd; hWnd = CreateWindow(// Create a win32 window APPTITLE, APPTITLE, WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, 500, 400, NULL, NULL, hInstance, NULL); if(!hWnd) return FALSE; ShowWindow(hWnd, nCmdShow); UpdateWindow(hWnd); return TRUE;} int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow){ MSG msg; MyRegisterClass(hInstance); if(!InitInstance(hInstance,nCmdShow)) return 1; while(GetMessage(&msg, NULL, 0, 0)) { TranslateMessage(&msg); DispatchMessage(&msg); } return msg.wParam;} In the Open event handler of the default window: Hello, world! Hello, world! The first paragraph of the W3C Recommendation on the global structure of an HTML document also features this example. This is the smallest legal version, leaving out all optional tagsHello, world! Hello, world! Hello, world! Hello, world! %PDF-1.01 0 obj<<>>endobj 2 0 objendobj 3 0 objendobj 4 0 obj/MediaBox [0 0 612 792]/Contents 5 0 R>>endobj5 0 obj>streamBT/F1 24 TF100 100 Td (Hello, world!) TjETendstreamendobj6 0 obj[/PDF /Text]endobj7 0 objendobjxref0 80000000000 65535 f0000000009 00000 n0000000074 00000 n0000000120 00000 n0000000179 00000 n0000000322 00000 n0000000415 00000 n0000000445 00000 ntrailerstartxref553%%EOFThis is a valid PDF only if the text file has CRLF line endings. % Displays on console. (Hello, world!) = %! % Displays as page output. /Courier findfont 24 scalefont setfont 100 100 moveto (Hello, world!) show showpage {\rtf1\ansi\deff0{\fonttbl {\f0 Courier New;}}\f0\fs20 Hello, world!} .+++o.l.+++++++\/ #/?\ $H!\++++++\ + \comma.---------- .>++++.< .+.@/.# \@@@@=>++++>++++ $Hello2$Output: $Hello2$CloseBuf: $Hello$Note: The first method is tested and it's working. I don't know about the second one! Decimal to Binary converter Binary calculator Binary number is a number expressed in the base 2 numeral system. Binary number's digits have 2 symbols: zero (0) and one (1). Each digit of a binary number counts a power of 2. Binary number example:11012 = 123+122+021+120 = 1310 Decimal Decimal number is a number expressed in the base 10 numeral system. Decimal number's digits have 10 symbols: 0,1,2,3,4,5,6,7,8,9. Each digit of a decimal number counts a power of 10. Decimal number example:65310 = 6102+5101+3100 How to convert binary to decimal For binary number with n digits: dn-1...d3d2d1d0 The decimal number is equal to the sum of binary digits (dn) times their power of 2 (2n): decimal = d020 + d121 + d222 + ... Example Find the decimal value of 1110012: binary number: 1 1 1 0 0 1 power of 2: 25 24 23 22 21 20 1110012 = 125+124+123+022+021+120 = 5710 Binary to decimal conversion table BinaryNumber DecimalNumber HexNumber 0 0 0 1 1 1 10 2 2 11 3 3 100 4 4 101 5 5 110 6 6 111 7 7 1000 8 8 1001 9 9 1010 10 A 1011 11 B 1100 12 C 1101 13 D 1110 14 E 1111 15 F 10000 16 10 10001 17 11 10010 18 12 10011 19 13 10100 20 14 10101 21 15 10110 22 16 10111 23 17 11000 24 18 11001 25 19 11010 26 1A 11011 27 1B 11100 28 1C 11101 29 1D 11110 30 1E 11111 31 1F 100000 32 20 1000000 64 40 10000000 128 80 100000000 256 100 See also * You can enter decimals with e notation. e.g: 572 = 5.72e2. Binary to Decimal conversion Binary calculator Decimal number is a number expressed in the base 10 numeral system. Decimal number's digits have 10 symbols: 0,1,2,3,4,5,6,7,8,9. Each digit of a decimal number counts a power of 10. Decimal number example:65310 = 6102+5101+3100 Binary Binary number is a number expressed in the base 2 numeral system. Binary number's digits have 2 symbols: zero (0) and one (1). Each digit of a binary number counts a power of 2. Binary number example:11012 = 123+122+021+120 = 1310 How to convert decimal to binary Conversion steps: Divide the number by 2. Get the integer quotient for the next iteration. Get the remainder for the binary digit. Repeat the steps until the quotient is equal to 0. Example #1 Convert 1310 to binary: Divisionby 2 Quotient Remainder Bit # 13/2 6 1 0 6/2 3 0 1 3/2 1 1 2 1/2 0 1 3 So 1310 = 11012 Example #2 Convert 17410 to binary: Divisionby 2 Quotient Remainder Bit # 174/2 87 0 0 87/2 43 1 1 43/2 21 1 2 21/2 10 1 3 10/2 5 0 4 5/2 2 1 5 2/2 1 0 6 1/2 0 1 7 So 17410 = 101011102 Decimal to binary conversion table DecimalNumber BinaryNumber HexNumber 0 0 0 1 1 1 2 10 2 3 11 3 4 100 4 5 101 5 6 110 6 7 111 7 8 1000 8 9 1001 9 10 1010 A 11 1011 B 12 1100 C 13 1101 D 14 1110 E 15 1111 F 16 10000 10 17 10001 11 18 10010 12 19 10011 13 20 10100 14 21 10101 15 22 10110 16 23 10111 17 24 11000 18 25 11001 19 26 11010 1A 27 11011 1B 28 11100 1C 29 11101 1D 30 11110 1E 31 11111 1F 32 100000 20 64 1000000 40 128 10000000 80 256 100000000 100 See also

Salon business plan examples. Salon business template. Business plan salon template. Beauty salon business plan template free pdf. Salon business plan pdf download. Nail salon business plan template pdf. Hair salon business plan example. How do i write a business plan for a salon. Salon business plan template doc. Salon business plan free.