Копипаста:Hello, world! — Lurkmore

ACHTUNG! Опасно для моска!

Министерство здравоохранения Луркмора предупреждает: вдумчивое чтение нижеследующего текста способно нанести непоправимый ущерб рассудку. Вас предупреждали.

«Hello, world!» на разных языках.

Anonymous

Hello world

```
/
 /<'Hello, world!/ | вывод + привет мир
\
```

Простенькая программка, складывающая два числа.

```
/
/int/x, y, z\/ | Целый тип + переменные.
/>/y/x/        | Поочерёдный ввод значений переменных
               |(для ввода в пределах одной строки />/y/x\/).
/z/x/+/y\/  |/z/x/+/y\/= "z := (x + y)", a /z/x/+/y/ = "z := x, + y"
/</x/y\/    |/</x/y\/ выводит всё в одной строке, а /</x/y/ в двух разных
/</z/
\
```

Весь исходный код может уместиться в пределах одной строки:

```
/
|Если писать код в пределах одной строки, образуются клещи (//), что недопустимо!
/<'H/'e/'l/'l/'o/',/' /'n/'e/'w/'f/'a/'g\/char/x/>/x/if/x/'hi\/<'win\/if/x/'wtf?\/</'The is manual for newfag\/else\/</'natribu.org\/
 |Равносильно
/<'H/'e/'l/'l/'o/',/' /'n/'e/'w/'f/'a/'g\/
/char/x/
/>/x/
/if/x/'hi\                      |если x = hi. после условия "/" не ставится
 /<'win\/                       |а вот после действий ставится
/if/x/'wtf?\                    |если x = wtf
 /<'The is manual for newfag\/
/else\                          |иначе
 /<'natribu.org\/
\
```

Ради наименьшего удобства в использовании, кол-во букв в названиях всех стандартных процедур, функций и типов не превышает 4 символов.

BAT

```
@echo off ;традиционная строка любого батника
echo Hello, World!
```

или так

```
@echo Hello, World!
```

BASIC

```
10 PRINT "Hello, World!"
```

Quick/Turbo BASIC:

```
? "Hello, World!"
```

InfoBASIC:

```
CRT 'Hello, World!'
```

PureBasic

Консоль:

```
OpenConsole()
PrintN("Hello, World!")
Input()
```

Диалоговое окно:

```
MessageRequester("Заголовок", "Hello, World!")
```

Окно:

```
OpenWindow(0, 0, 0, 200, 50, "Заголовок", #PB_Window_MinimizeGadget|#PB_Window_ScreenCentered)
TextGadget(0, 10, 16, 180, 16,"Hello, World!", #PB_Text_Center)
Repeat
  Event = WaitWindowEvent()
Until Event = #PB_Event_CloseWindow
```

Fortran 77

На культовом:

```
      PROGRAM HELLOW
      WRITE(UNIT=*, FMT=*) 'Hello World'
      END
```

Ruby

Почти не отличается:

```
puts 'Hello, World!'
```

Pawn

```
print("Hello,World!");
```

Python

Похожим образом:

```
print "Hello, World!"
```

…или так:

```
import __hello__
```

А вот в **Python** 3.0 (aka Py3k, Пузик) — немного по-другому:

```
print("Hello, World!")
```

MS-DOS shell

```
echo Hello, world!
```

Rexx

```
Say 'Hello, world!'
```

Pascal

```
begin
  write('Hello, World!')
end.
```

Delphi

```
{$apptype console}
begin
  Writeln('Hello, World!');
end.
```

На WinAPI:

```
uses windows;
begin
  MessageBox(0,'Hello, World!','Заголовок',0);
end.
```

На ООП:

```
uses windows;
{$apptype console}
Type
  THelloWorld = class
  procedure ShowMsg();
  procedure PrintMsg();
  end;
Procedure THelloWord.ShowMsg();
begin
  MessageBox(0,'Hello, World!','Заголовок',0);
end;

Procedure THelloWord.PrintMsg();
begin
  WriteLn('Hello, World!');
end;

var
 H : THelloWorld;
begin
 H := THelloWorld.Create;
 H.ShowMsg;
 H.PrintMsg;
end.
```

Ада

```
with Ada.Text_IO;
procedure Hello_World is
begin
  Ada.Text_IO.Put_Line ("Hello World");
end Hello_World;
```

Модула-3

```
MODULE Main;
IMPORT IO;
BEGIN
  IO.Put ("Hello World\n")
END Main.
```

Visual Basic

```
Sub Main()
   print "Hello, World!"
End Sub
```

А в VBA по-другому, но с окном

```
Sub Main()
   MsgBox "Hello, World!"
End Sub
```

Lotus Script

```
  Messagebox "Hello, World!", MB_OK
```

Или так (не самый заметный вариант):

```
  Print "Hello, World!"
```

Или рядом с LS:

```
  @Prompt([OK];"";"Hello, World!");
```

Ещё на "собаках" (аналог print'а):

```
  @StatusBar("Hello, World!")
```

InstallScript

```
  Messagebox("Hello, World!", MB_OK);
```

AutoIT

```
  MsgBox(0, "AutoIT Window", "Hello, world!", 0.75)
```

C

```
#include <stdio.h>
```

```
int main(void)
{
  printf("Hello, World!\n");
  return 0;
}
```

Алсо, совместимо с C++.

Objective-C для Яблочников и GNUstep'a

```
#import <Foundation/Foundation.h>
@interface HelloWorld : NSObject
@end
@imlementation HelloWorld
+load{
  NSLog(@"Hello, World!");
  return;
}
@end
```

C++

```
#include <iostream>
int main()
{
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Чуть более извращённый вариант:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, World!" << endl;

    return 0;
}
```

Win32 API

```
#include <windows.h>
int WINAPI WinMain (HINSTANCE,HINSTANCE,PSTR,int){
MessageBox(0,"Hello, World","F*ck",0);
return 0;
}
```

C++ ООП

```
#include <iostream>

class World
{
    public:
        static int Hello()
        {
            std::cout << "Hello, world!" << std::endl;
            return 0;
        }
};

int main()
{
    return World::Hello();
}
```

С шаблонами:

```
class World{};

template <typename T>
void Hello(T hello)
{
    std::cout << "Hello, " << hello << std::endl;
}

template <class World>
void Hello()
{
    std::cout << "Hello, world!" << std::endl;
}

int main()
{
    Hello<World>();
}
```

Чудеса К3 грамматики:

```
template <typename T>
class Hello
{
    public:
        Hello(T hello)
        {
            std::cout << "Hello, " << hello << std::endl;
        }
};

template <>
class Hello<class World>
{
    public:
        Hello()
        {
            std::cout << "Hello, world!" << std::endl;
        }
};

int main()
{
    Hello<World>();
}
```

С наследование

```
#include <iostream>

class AbstractHello
{
    public:
        virtual ~AbstractHello(){std::cout << " World!";}
        void Prnt(){std::cout << "Hello";}
```

```
};

class ChildHello: public AbstractHello
{
    public:
        ~ChildHello(){Prnt();}
};

int main()
{
    ChildHello *Obj;
    Obj = new ChildHello;
    delete Obj;
}
```

Просто, чтобы повыёбываться (паттерны)

```
/*!
 * Hello world! application
 *
 * \file hello.cpp
 */

#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>
#include <cassert>

/*!
 * Dispay message.
 */
void displayMessage();

/*!
 * Sentence type
 *
 * Type of sentence, used to decide how to terminate sentence.
 */
enum ESentenceType {
    eStatement,
    eExclamation,
    sQuestion,
    eCommand
};

/*!
 * Utility class to prevent unintended copying of class instances.
 */
class nonCopyable {
protected:
    nonCopyable() {
    }

    ~nonCopyable() {
    }

private:
    nonCopyable(const nonCopyable&);
    const nonCopyable& operator=(const nonCopyable&);
};

/*!
 * Utility function to obtain punctuation mark to end sentence
 * of specified type.
 */
inline char getPunctionMark(ESentenceType sentenceType) {
    char puncMark = '.';
    switch(sentenceType) {
        case eStatement  : puncMark = '.'; break;
        case eExclamation: puncMark = '!'; break;
        case sQuestion   : puncMark = '?'; break;
        case eCommand    : puncMark = '.'; break;
        default: {
            // should never get here
            assert(false);
        }
    }
    return puncMark;
}

/*!
 * Utility class for creation of instances.
 */
template<typename TElem>
class Creatable {
protected:
    Creatable() {
    }

    virtual ~Creatable() {
        clear();
    }

public:
    static TElem* create() {
        TElem* e = new TElem;
        return e;
    }

    void free() {
        delete this;
    }

    virtual void clear() {
    }
};

template<typename TElem, typename TParam>
class CreatableParam {
protected:
    CreatableParam() {
    }

    virtual ~CreatableParam() {
    }

public:
    static TElem* create(TParam p) {
        TElem* e = new TElem;
        e->initialize(p);
        return e;
    }

    void free() {
        finalize();
        delete this;
    }
```

```
    virtual void initialize(TParam /*p*/) {
    }

    virtual void finalize() {
        clear();
    }

    virtual void clear() {
    }
};

/*!
 * Base class for displayable content
 */
class DisplayElem
: public nonCopyable {
protected:
    DisplayElem() {
    }

    virtual ~DisplayElem() {
    }

public:
    virtual void display(std::ostream& os) const = 0;
};

/*!
 * STL algorithm for displaying elements
 */
class Displayer
: public std::unary_function<void, const DisplayElem*> {
private:
    std::ostream& m_os;
    char   m_sep;
    size_t m_count;

public:
    Displayer(std::ostream& os, char sep = '\0')
    : m_os(os)
    , m_sep(sep)
    , m_count(0) {
    }

    ~Displayer() {
    }

    void operator()(const DisplayElem* e) {
        if(('\0' != m_sep) && (0 < m_count)) {
            m_os << m_sep;
        }
        e->display(m_os);
        ++m_count;
    }
};

/*!
 * STL algorithm for freeing display elements
 */
template <typename TElem>
class Freer
: public std::unary_function<void, TElem*> {
public:
    void operator()(TElem* e) {
        e->free();
    }
};

/*!
 * Display element for letter.
 *
 * The letter is the fundamental element: it has no substructure.
 */
class Letter
: public DisplayElem
, public CreatableParam<Letter, char> {
private:
    char m_ch;

protected:
    /*virtual*/ ~Letter() {
    }

public:
    Letter() : m_ch('\0') {
    }

    void initialize(char ch) {
        m_ch = ch;
    }

    void finalize() {
        m_ch = '\0';
    }

    void display(std::ostream& os) const {
        os << m_ch;
        // no endLetter()
    }
};

/*!
 * Display element for word.
 *
 * A word is a sequence of letters.
 */
class Word
: public DisplayElem
, public Creatable<Word> {
private:
    std::vector<Letter*> m_letters;

protected:
    /*virtual*/ ~Word() {
        clear();
    }

public:
    Word() {
    }

    void clear() {
        std::for_each(m_letters.begin(), m_letters.end(), Freer<Letter>());
        m_letters.clear();
    }

    void addLetter(Letter* s) {
```

```
        m_letters.push_back(s);
    }

    /*virtual*/ void display(std::ostream& os) const {
        std::for_each(m_letters.begin(), m_letters.end(), Displayer(os));
        // no endLetter()
    }
};

/*!
 * Display element for sentence.
 *
 * A sentence is a sequence of words.
 */
class Sentence
: public DisplayElem
, public CreatableParam<Sentence, ESentenceType> {
private:
    std::vector<Word*> m_words;

    ESentenceType m_sentenceType;

protected:
    /*virtual*/ ~Sentence() {
        clear();
    }

    void endSentence(std::ostream& os) const {
        const char puncMark = getPunctionMark(m_sentenceType);
        os << puncMark;
    }

public:
    Sentence()
    : m_sentenceType(eStatement) {
    }

    void initialize(ESentenceType sentenceType) {
        m_sentenceType = sentenceType;
    }

    void finalize() {
        m_sentenceType = eStatement;
    }

    void clear() {
        std::for_each(m_words.begin(), m_words.end(), Freer<Word>());
        m_words.clear();
    }

    void addWord(Word* w) {
        m_words.push_back(w);
    }

    void display(std::ostream& os) const {
        std::for_each(m_words.begin(), m_words.end(), Displayer(os, ' '));
        endSentence(os);
    }
};

/*!
 * Display element for message.
 *
 * A message is a sequence of sentences.
 */
class Message
: public DisplayElem
, public Creatable<Message> {
private:
    std::vector<Sentence*> m_sentences;

protected:
    /*virtual*/ ~Message() {
        clear();
    }

    void endMessage(std::ostream& os) const {
        os << std::endl;
    }

public:
    Message() {
    }

    void clear() {
        std::for_each(m_sentences.begin(), m_sentences.end(), Freer<Sentence>());
        m_sentences.clear();
    }

    void addSentence(Sentence* s) {
        m_sentences.push_back(s);
    }

    void display(std::ostream& os) const {
        std::for_each(m_sentences.begin(), m_sentences.end(), Displayer(os, ' '));
        endMessage(os);
    }
};

/*!
 * Main entrance point.
 */
int main() {
    displayMessage();
    return 0;
}

/*!
 * Display message.
 */
void displayMessage() {
    Word* first_word = Word::create();
    first_word->addLetter(Letter::create('H'));
    first_word->addLetter(Letter::create('e'));
    first_word->addLetter(Letter::create('l'));
    first_word->addLetter(Letter::create('l'));
    first_word->addLetter(Letter::create('o'));

    Word* second_word = Word::create();
    second_word->addLetter(Letter::create('w'));
    second_word->addLetter(Letter::create('o'));
    second_word->addLetter(Letter::create('r'));
    second_word->addLetter(Letter::create('l'));
    second_word->addLetter(Letter::create('d'));

    Sentence* sentence = Sentence::create(eExclamation);
    sentence->addWord(first_word);
    sentence->addWord(second_word);
```

```
    Message* message = Message::create();
    message->addSentence(sentence);

    message->display(std::cout);

    message->free();
    // sentences, etc freed by parent
}
```

C#

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Hello, World!");
    }
}
```

или так

```
class Program
{
    static void Main()
    {
        System.Windows.Forms.MessageBox.Show("Hello, World!");
    }
}
```

или даже так, если версия C# >=9.0

```
System.Console.Write("Hello, World!");
```

Или с использованием DI-контейнеров

```
using System;
using Microsoft.Practices.Unity;

class Program
{
    static void Main(string[] args)
    {
        var writer = new ConsoleMessageWriter();
        UnityContainer uc = new UnityContainer();
        uc.RegisterType<IMessageWriter, ConsoleMessageWriter>(new ContainerControlledLifetimeManager());
        var salutation = uc.Resolve<Salutation>();

        salutation.Exclaim();
    }
}

public interface IMessageWriter
{
    void Write(string message);
}

public class ConsoleMessageWriter : IMessageWriter
{
    public void Write(string message)
    {
        Console.WriteLine(message);
    }
}

public class Salutation
{
    [Dependency]
    public IMessageWriter Writer { get; set; }

    public Salutation() {}

    public void Exclaim()
    {
        Writer.Write("Hello world");
    }
}
```

F#

```
printfn "Hello, World!"
```

Boo

```
print("Hello, World!")
```

Или:

```
print "Hello, World!"
```

Или даже так:

```
System.Console.WriteLine("Hello, World!")
```

Perl

```
#!/usr/bin/perl
print "Hello, World!\n";
```

или так:

```
perl -le "print 'Hello, World!';"
```

или вообще так:

```
perl -e 'use feature q/say/; say "Hello, World!"'
```

Haskell

Так — в весьма простом и довольно красивом, но малопопулярном языке Haskell (обратите внимание на прекрасную читаемость и простоту кода):

```
{-# LANGUAGE NoImplicitPrelude #-}

import qualified Data.ByteString as BS -- бесполезный импорт. Помог бы автору, но мне тоже лень.
import Data.Foldable (foldl1)
import Data.Function ((.))
import Text.Show (show)
import Data.List ((++))
import Data.Char (Char)
import Control.Monad ((>>=))
import System.IO
```

```
main :: IO ()
       -- а тут автор был настолько ленив, что использовал один приём 2 раза. Подряд.
main = hSetBuffering stdout NoBuffering >>= \ _ -> hSetBuffering stdin LineBuffering >>= \ _ -> putHelloWorld (message1)
  where
    message :: [[Char]]
    message = [ 'H':'e':'l':'l':'o':',':[], 'w':'o':'r':'l':'d':'!':[] ]

    putHelloWorld :: [Char] -> IO () -- putHelloWorld = putStrLn
    putHelloWorld []     = System.IO.hPutStr stdout ('\n':[])
    putHelloWorld (x:xs) = System.IO.hPutStr stdout (x:[]) >>= \ _ -> putHelloWorld xs

    message1 = let f = (++) . (++ " "); f1 = foldl1 f
               in  f1 message
```

Чуть менее полный ~~матана~~ говнокода вариант на Haskell:

```
main = putStrLn "Hello, world!"
```

LISP

Вообще-то будет так:

```
(eval (cons (quote mapcar)
            (cons (cons (quote function)
                        (cons (quote princ) ()))
                  (cons (cons (quote quote)
                              (cons (cons #\H (cons #\e (cons #\l (cons #\l (cons #\o
                                    (cons #\, (cons #\Space
                                     (cons #\w (cons #\o (cons #\r (cons #\l (cons #\d (cons #\!
                                      ()))))))))))))) ())) ())))
```

Но можно и так:

```
(mapcar #'princ '(#\H #\e #\l #\l #\o #\, #\Space #\w #\o #\r #\l #\d #\!))
```

Или так:

```
(princ "Hello, world!")
```

Алсо в Common Lisp:

```
(format t "hello world~%")
```

Или даже так (в REPL):

```
"Hello, world!"
```

Clojure

```
(println "Hello world")
```

BrainFuck

Нечитаемый вариант:

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.
```

Китайский стиль:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++.+++++++++++++++++++++++++++++.+++++++..+++.-----------------------------------------------------------------------------
```

Forth

```
: HELLO 72 EMIT 101 EMIT 2 0 DO 108 EMIT LOOP 111 EMIT ;
: WORLD 119 EMIT 111 EMIT 114 EMIT 108 EMIT 100 EMIT ;
: HELLO_WORLD HELLO 44 EMIT SPACE WORLD 33 EMIT ;
HELLO_WORLD
```

Тот же Forth без выебоновЪ:

```
: HelloWorld ." Hello, world!" ;
```

Factor

```
USE: io
IN: hello-world

: hello ( -- ) "Hello, world!" print ;
MAIN: hello
```

1С:Предприятие 7.7/8/8.1/8.2

Так решается эта задача во встроеном языке отечественного производителя 1С:

```
Предупреждение("Hello, world");
```

Как вариант:

```
DoMessageBox("Hello, world",,"Attention");
```

Или так:

```
Сообщить("Привет, Мир!");
```

Или так:

```
Message("Привет, Мир!", MessageStatus.Attention);
```

А на 8.2 еще и так можно:

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "Хелло, Ворлд";
Сообщение.Сообщить();
```

И вот так тоже можно:

```
Message = New User Message;
Message.Text = "Хелло, Ворлд";
Message.Message();
```

ну и вот так еще

```
Вопрос("Хелло, Ворлд!", РежимДиалогаВопрос.ОК);
```

Java

Кофеиновый код:

```
public class HelloWorld {
```

```
  public static void main(String[] args) {
    System.out.println("Hello, world!");
  }
}
```

С выебонами:

```
//hello class
public class HelloWorld extends java.lang.Object {

    //args - arguments from cmd
    public static final void main(java.lang.String[] args) {

        java.lang.System.out.println(new String("Hello, World!"));
    }
}
```

Та же жаба, с выебонами, зависящая от старого багованного верификатора кода.

```
public class HelloWorld {
  static
  {
    System.out.println("Hello, world!");
    System.exit(0);
  }
}
```

Жаба с окном

```
public class hw {
public static void main (String [] args){
javax.swing.JOptionPane.showMessageDialog(null,"Хэлловорлд!");
}
}
```

Жаба с окном, сделанным своими руками

```
import java.awt.*;
import javax.swing.*;
public class HelloWorld extends JFrame{
        HelloWorld(){
                setVisible (true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setSize(300,300);
                Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
                setLocation(dim.width/2-getSize().width/2,dim.height/2-getSize().width/2);
                JLabel l = new JLabel("Хэловорлд");
                setLayout(null);
                l.setBounds(110, 110, 70, 20);
                add(l);
        }
        public static void main(String[] args) {
                new HelloWorld();
        }
}
```

Выебоны с паттернами

```
public interface Subject {
    public void attach(Observer observer);
    public void detach(Observer observer);
    public void notifyObservers();
}

public interface Observer {
    public void update(Subject subject);
}

public class HelloWorldSubject implements Subject {

    private ArrayList<Observer> observers;
    private String str;

    public HelloWorldSubject() {
        super();

        observers = new ArrayList<Observer>();
    }

    public void attach(Observer observer) {
        observers.add(observer);
    }

    public void detach(Observer observer) {
        observers.remove(observer);
    }

    public void notifyObservers() {
        Iterator<Observer> iter = observers.iterator();

        while (iter.hasNext()) {
            Observer observer = iter.next();
            observer.update(this);
        }
    }

    public String getStr() {
        return str;
    }

    public void setStr(String str) {
        this.str = str;
        notifyObservers();
    }
}

public class HelloWorldObserver implements Observer {

    public void update(Subject subject) {
        HelloWorldSubject sub = (HelloWorldSubject)subject;
        System.out.println(sub.getStr());
    }

}

public interface Command {
    void execute();
}

public class HelloWorldCommand implements Command {

    private HelloWorldSubject subject;

    public HelloWorldCommand(Subject subject) {
        super();

        this.subject = (HelloWorldSubject)subject;
    }
```

```
    public void execute() {
        subject.setStr("hello world");
    }

}

public interface AbstractFactory {
    public Subject createSubject();
    public Observer createObserver();
    public Command createCommand(Subject subject);
}

public class HelloWorldFactory implements AbstractFactory {

    public Subject createSubject() {
        return new HelloWorldSubject();
    }

    public Observer createObserver() {
        return new HelloWorldObserver();
    }

    public Command createCommand(Subject subject) {
        return new HelloWorldCommand(subject);
    }
}

public class FactoryMakerSingleton {

    private static FactoryMakerSingleton instance = null;
    private AbstractFactory factory;

    private FactoryMakerSingleton() {
        factory = new HelloWorldFactory();
    }

    public static synchronized FactoryMakerSingleton getInstance() {
        if (instance == null) {
            instance = new FactoryMakerSingleton();
        }

        return instance;
    }

    public AbstractFactory getFactory() {
        return factory;
    }
}

public class AbuseDesignPatterns {

    public static void main(String[] args) {
        AbstractFactory factory = FactoryMakerSingleton.getInstance().getFactory();

        Subject subject = factory.createSubject();
        subject.attach(factory.createObserver());

        Command command = factory.createCommand(subject);

        command.execute();
    }

}
```

Scala

```
object HelloWorld extends App {
  println("Hello, world!")
}
```

ActionScript 2.0

```
trace ("hello world!");
```

ActionScript 3.0

```
trace ("hello world!");
```

Или так:

```
var field:TextField=new TextField();
field.text="hello world!";
addChild(field);
```

Eiffel

```
class HELLO_WORLD
feature
  print_hello is
      -- Print "Hello, World!"
    do
      print ("Hello, World!")
    end
end
```

MIDlet Pascal

```
Program Hello;
Begin
  DrawText('Hello, world!', 5, 5);
  Repaint; Delay(5000);
End.
```

PHP

```
echo "Hello, world!";
```

Или так (при вставке кода в HTML):

```
<?="Hello, world!"?>
```

Или с использованием ОО-паттернов программирования:

```
/*************************************************************************
Model-View-Controller implementation according to POSA
(Pattern-Oriented Software Architecture
  http://www.hillside.net/patterns/books/Siemens/book.html)
*************************************************************************/

class HelloWorldController {
    private $model;
```

```
    function __construct($model) {
        $this->model = $model;
    }

    function handleEvent($args) {
        $this->model->setStrategy($args[2]);
        $this->model->addText($args[1]);
    }
}


class HelloWorldModel {
    private $text;
    private $observers = array();
    private $strategy;

    function attach($observer) {
        $this->observers[] = $observer;
    }

    function getData() {
        $facade = new HelloWorldFacade($this->strategy);
        return $facade->getHelloWorld().$this->text."\n";
    }

    function addText($text='') {
        $this->text = $text;
        $this->notify();
    }

    function setStrategy($strategy) {
        $this->strategy = $strategy;
    }

    function notify() {
        foreach ($this->observers as $observer) {
            $observer->update();
        }
    }
}

class HelloWorldView {
    private $model;

    function initialize($model) {
        $this->model = $model;
        $model->attach($this);
        return $this->makeController();
    }

    function makeController() {
        return new HelloWorldController($this->model);
    }

    function update() {
        $this->display();
    }

    function display() {
        echo $this->model->getData();
    }
}


/**********************************************************************
"Business logic"
**********************************************************************/

class HelloWorld {
   function execute() {
       return "Hello world";
   }
}

class HelloWorldDecorator {
   private $helloworld;
   function __construct($helloworld) {
       $this->helloworld = $helloworld;
   }

   function execute() {
       return $this->helloworld->execute();
   }
}

abstract class HelloWorldEmphasisStrategy {
    abstract function emphasize($string);
}

class HelloWorldBangEmphasisStrategy extends HelloWorldEmphasisStrategy {
    function emphasize($string) {
        return $string."!";
    }
}

class HelloWorldRepetitionEmphasisStrategy extends HelloWorldEmphasisStrategy {
    function emphasize($string) {
        return $string." and ".$string." again";
    }
}

class HelloWorldEmphasizer extends HelloWorldDecorator {
   private $strategy;
   function HelloWorldEmphasizer($helloworld,$strategy) {
       $this->strategy = $strategy;
       parent::__construct($helloworld);
   }

   function execute() {
       $string = parent::execute();
       return $this->strategy->emphasize($string);
   }
}

class HelloWorldStrategyFactory {
    static function make($type) {
        if ($type == 'repetition') return self::makeRepetitionStrategy();
        return self::makeBangStrategy();
    }

    static function makeBangStrategy() {
        return new HelloWorldBangEmphasisStrategy;
    }
    static function makeRepetitionStrategy() {
        return new HelloWorldRepetitionEmphasisStrategy;
    }
}
```

```
class HelloWorldFormatter extends HelloWorldDecorator {
   function execute() {
       $string = parent::execute();
       return $string."\n";
   }
}

class HelloWorldFacade {
    private $strategy;
    function __construct($strategyType) {
        $this->strategy = HelloWorldStrategyFactory::make($strategyType);
    }

    function getHelloWorld() {
        $formatter = new HelloWorldFormatter(
                new HelloWorldEmphasizer(
                    new HelloWorld,$this->strategy));
        return $formatter->execute();
    }
}

$model = new HelloWorldModel;
$view = new HelloWorldView;
$controller = $view->initialize($model);
$controller->handleEvent($_SERVER['argv']);
```

Vbscript

```
Msgbox "Hello world!"
```

Javascript

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$
```

Или вот так (не работает, когда броузер парсит страницу как XHTML, то есть в далеком будущем, когда сервера начнут отдавать страницы с MIME-типом application/xhtml+xml)

```
document.write("Hello, world!");
```

Или так ("классы")

```
(function() {
   function Chatterbox(msg) {
      this.msg = msg || 'Hello, world!';
   }

   Chatterbox.prototype.greet = function() {
      alert(this.msg);
   }

   var chatter = new Chatterbox();
   chatter.greet();
})();
```

Или вот так (окно с сообщением)

```
alert("Hello, world!");
```

В NodeJS (распечатает в командную строку) а в браузере(в отладчик в консоль):

```
console.log("Hello, world!");
```

Или вот так на **Action script 3.0** (флэшовая реализация стандарта ECMAScript):

```
trace('Hello World!');

//Или, если непременно надо вывести прямо на экран, а не в консоль, так:
var tf:TextField = new TextField;
tf.text = "Hello World!";
addChild(tf);
```

Или если ООП:

```
package {
  import flash.display.Sprite;

  public class HelloWorld extends Sprite {
    public function HelloWorld() {
      trace ("Hello, world!");
    }
  }
}
```

Теперь оно и сервер-сайд!

```
#!/usr/bin/env node
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

sh

```
echo Hello, world!
```

Или так

```
cat <<EOF
Hello world!
EOF
```

Индусский вариант

```
for i in 'H' 'e' 'l' 'l' 'o' ' ' 'w' 'o' 'r' 'l' 'd' '!' '\n'; do
        echo -ne "$i"
done
```

На bash.orge

http://bash.org.ru/?text=Hello+world

На Прологе

```
goal :-
  write ("Hello, World!").
```

SQL запрос

SQL-92

```
select 'Hello, World!'
```

Oracle SQL

```
select 'Hello, World!' from dual;
```

Oracle PL/SQL

```
SET serveroutput ON

BEGIN
 dbms_output.put_line('Hello world!');
END;
/
```

PostgreSQL

```
select 'Hello, World!';
```

PostgreSQL PL/pgSQL

```
DO $$
BEGIN
   RAISE NOTICE 'Hello world!';
END;
$$;
/
```

T-SQL (Microsoft SQL Server)

```
PRINT 'Hello, world';
```

MySQL

```
select 'Hello, World!';
```

FireBird SQL / InterBase

```
select 'Hello, World!' FROM RDB$DATABASE;
```

Informix

```
SELECT FIRST 1 'Hello, World!' FROM systables;
```

Обычно хеллоуворлдщика можно ввести в транс, добавив в какой-нибудь частоиспользуемый header-файл следующие строчки (ахтунг, C-specific!):

```
#ifndef DEFINE_ME
#error Fatal error! There must be some brain in your head!
#endif
```

Ассемблер

Очевидно, что никакой сложности в решении такая задача собой не представляет. Тем не менее, решив подобную задачу на каком-либо языке программирования, субъект чаще всего начинает *ошибочно* самоидентифицироваться с программистом.

Однако на языках ассемблера данная задача представляется более сложной:

Assembler i8086, MS-DOS, fasm

```
use16
org 100h

mov ah,09h
mov dx,msg
int 21h

mov ax,4C00h
int 21h
msg db 'Hello, World!$'
```

Assembler i8086, MS-DOS, masm

```
.model tiny
.code
org 100h

Start:
mov ah, 9
mov dx, offset msg
int 21h

mov ax, 4C00H
int 21h

msg db 'Hello, world$'
end Start
```

Assembler i8086, MS-DOS, tasm

```
mov ax, 0b800h
mov ds, ax
mov [02h], 'H'
mov [04h], 'e'
mov [06h], 'l'
mov [08h], 'l'
mov [0ah], 'o'
mov [0ch], ','
mov [0eh], 'W'
mov [10h], 'o'
mov [12h], 'r'
mov [14h], 'l'
mov [16h], 'd'
mov [18h], '!'
ret
```

Assembler i386, Linux, nasm

```
SECTION .data
msg: db "Hello, world",10
len: equ $-msg

SECTION .text
global main
main:
mov edx, len
mov ecx, msg
mov ebx, 1
mov eax, 4
int 0x80
```

```
mov ebx, 0
mov eax, 1
int 0x80
```

То же, только GAS

```
.data
msg: .ascii "Hello,world!\n"
len = . - msg

.text
.globl _start
_start:
movl $4,%eax
movl $1,%ebx
movl $msg,%ecx
movl $len,%edx
int $0x80

xorl %ebx,%ebx
movl $1,%eax
int $0x80
```

AMD64 Linux, GAS

```
.text
prefix: .ascii "Hello, World\n"
.set prefix_size, . - prefix
.global _start
_start:
xorq %rax, %rax
incb %al
xorl %edi, %edi
incl %edi
movq $prefix, %rsi
mov $prefix_size, %edx
syscall
movl $60, %eax
xorl %edi, %edi
syscall
```

Assembler i386, Windows, masm

```
.386
.model flat, stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib

.data
msg db "Hello, world!", 13, 10
len equ $-msg

.data?
written dd ?

.code
start:
push -11
call GetStdHandle

push 0
push offset written
push len
push offset msg
push eax
call WriteFile

push 0
call ExitProcess
end start
```

Или так:

```
.386
.model flat, STDCALL
includelib kernel32.lib
GetStdHandle PROTO:DWORD
WriteConsoleA PROTO:DWORD,:DWORD,:DWORD,:DWORD,:DWORD
ExitProcess PROTO:DWORD
.const
message db "Hello, world!"
.code
Main PROC
LOCAL hStdOut :DWORD
push -11
call GetStdHandle
mov hStdOut,EAX
push 0
push 0
push 16d
push offset message
push hStdOut
call WriteConsoleA
push 0
call ExitProcess
Main ENDP
End Main
```

Assembler fasm for Windows

```
include 'win32ax.inc'
s:
invoke MessageBox,0,'Хэловорлд','  ',0
.end s
```

Assembler микроконтроллер ATMega16, AVR Studio

```
.include "m16def.inc"
.cseg
.org $0000
rjmp  start      ;Reset handler
.org $0030
start:
ldi   r24,  25  ; ~= 9600 @ 4Mhz clock
out   UBRRL,  r24
out   UBRRH,  r2
ldi   r24,  1 << TXEN
out   UCSRB,  r24
ldi   r24,  1 << URSEL | 1 << UCSZ0 | 1 << UCSZ1 ; 8-n-1
out   UCSRC,  r24

; send msg
ldi   ZL, msg << 1
```

```
loop:
lpm   r0, Z+ ; next char
tst   r0   ; terminated?
stop: breq  stop

while_busy:
sbis  UCSRA,  UDRE
rjmp  while_busy
out   UDR,  r0
rjmp  loop
msg:  .db  "Hello, world!", 13, 10, 0
```

Assembler TASM для ZX Spectrum 48/128

```
  ORG #C000
Start:
  LD HL,Text
Loop:
  LD A,(HL)
  XOR A
  RET Z
  RST 16
  INC HL
  JR Loop
Text:
  DB 'Hello, world!
  DB 0
```

Машинный код Win32

Это можно вставить в шестнадцатеричный редактор, сохранить с расширением .exe, и оно сразу станет запускаемым под виндой, без всяких компиляторов и ассемблеров.

```
4D5A00000000000000000000001A1100002810000000000000000000000000000002F11000030100000211100000000000034110000000000000000000040000000504500004C0101000000000000000000000000000070000F010B0100000000000
```

Ниже представлена чуть более читабельная версия. В ASCII-представлении даже можно увидеть что-то человеческое, в частности, выводимую строку, названия функций WinAPI и названия библиотек, в которых они находятся. Сам исполняемый код находится по смещению 0xF0 и занимает 42 байта, включая строку "Hello, world!\0". А вообще, образ построен не по правилам.

```
Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F

00000000   4D 5A 00 00 00 00 00 00  00 00 00 00 1A 11 00 00   MZ
00000010   28 10 00 00 00 00 00 00  00 00 00 00 00 00 00 00   (
00000020   2F 11 00 00 30 10 00 00  21 11 00 00 00 00 00 00   /   0   !
00000030   34 11 00 00 00 00 00 00  00 00 00 00 40 00 00 00   4           @
00000040   50 45 00 00 4C 01 01 00  00 00 00 00 00 00 00 00   PE  L
00000050   00 00 00 00 70 00 0F 01  0B 01 00 00 00 00 00 00       p
00000060   00 00 00 00 00 00 00 00  F0 10 00 00 00 10 00 00           ?
00000070   00 00 00 00 00 00 40 00  00 10 00 00 00 02 00 00         @
00000080   00 00 00 00 00 00 00 00  03 00 0A 00 00 00 00 00
00000090   00 20 00 00 D9 00 00 00  00 00 00 00 02 00 00 00       U
000000A0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000B0   00 00 00 00 02 00 00 00  00 00 00 00 00 00 00 00
000000C0   00 10 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000D0   00 00 00 00 00 10 00 00  01 00 00 00 01 00 00 00
000000E0   00 00 00 00 00 00 00 00  00 00 00 00 60 00 00 E0                `   a
000000F0   6A 00 68 42 10 40 00 E8  0E 00 00 00 48 65 6C 6C   j hB @ e    Hell
00000100   6F 2C 20 77 6F 72 6C 64  21 00 6A 00 FF 15 30 10   o, world! j y 0
00000110   40 00 6A 00 FF 15 28 10  40 00 4B 45 52 4E 45 4C   @ j y ( @ KERNEL
00000120   33 32 00 45 78 69 74 50  72 6F 63 65 73 73 00 55   32 ExitProcess U
00000130   53 45 52 33 32 00 4D 65  73 73 61 67 65 42 6F 78   SER32 MessageBox
00000140   41 00 00 00 00 00 00 00                            A
```

На двух семисегментных индикаторах и VHDL

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY CTL IS
PORT(
        CLK:    IN STD_LOGIC;
        DOUT1:  OUT STD_LOGIC_VECTOR(0 TO 6);
        DOUT2:  OUT STD_LOGIC_VECTOR(0 TO 6)
);
END ENTITY;

ARCHITECTURE CTL_ARCH OF CTL IS
SIGNAL CNT:             STD_LOGIC_VECTOR(0 TO 3):="0000";

BEGIN
        PROCESS(CLK)
        BEGIN
                IF (CLK'EVENT AND CLK='1') THEN
                                IF CNT = "1011" THEN CNT <= "0000";
                                END IF;
                                CASE CNT IS
                                  WHEN "0000" => DOUT1 <= "1001000"; DOUT2 <="1111111";--H
                                  WHEN "0001" => DOUT1 <= "0110001"; DOUT2 <="1111111";--E
                                  WHEN "0010" => DOUT1 <= "1110001"; DOUT2 <="1111111";--L
                                  WHEN "0011" => DOUT1 <= "1110001"; DOUT2 <="1111111";--L
                                  WHEN "0100" => DOUT1 <= "0000001"; DOUT2 <="1111111";--O
                                  WHEN "0101" => DOUT1 <= "1111111"; DOUT2 <="1111111";--
                                  WHEN "0110" => DOUT1 <= "1100001"; DOUT2 <="1000001"; --W(1) W(2)
                                  WHEN "0111" => DOUT1 <= "0000001"; DOUT2 <="1111111";--O
                                  WHEN "1000" => DOUT1 <= "0011000"; DOUT2 <="1111111";--R
                                  WHEN "1001" => DOUT1 <= "1110001"; DOUT2 <="1111111";--L
                                  WHEN "1010" => DOUT1 <= "0000001"; DOUT2 <="1111111";--D
                                  WHEN OTHERS => DOUT1 <= "1111111";  DOUT2 <="1111111";
                                END CASE;
                                CNT<= CNT+1;
                END IF;
        END PROCESS;
END CTL_ARCH;
```

HQ9+ / HQ9++ / HQ9^{+}/_

```
H
```

C++ с использованием Component Object Model

На мелкомягком быдлоподелии:

```
[
    uuid(2573F8F4-CFEE-101A-9A9F-00AA00342820)
]
library LHello
{
    importlib("actimp.tlb");
    importlib("actexp.tlb");
#include "pshlo.idl"
    [
```

```
        uuid(2573F8F5-CFEE-101A-9A9F-00AA00342820)
    ]
    cotype THello
    {
        interface IHello;
        interface IPersistFile;
    };
};
[
    exe,
    uuid(2573F890-CFEE-101A-9A9F-00AA00342820)
]
module CHelloLib
{
    importheader();
    importheader();
    importheader();
    importheader("pshlo.h");
    importheader("shlo.hxx");
    importheader("mycls.hxx");
    importlib("actimp.tlb");
    importlib("actexp.tlb");
    importlib("thlo.tlb");
    [
        uuid(2573F891-CFEE-101A-9A9F-00AA00342820),
        aggregatable
    ]
    coclass CHello
    {
        cotype THello;
    };
};
```

```
#include "ipfix.hxx"
extern HANDLE hEvent;
class CHello : public CHelloBase
{
public:
        IPFIX(CLSID_CHello);
        CHello(IUnknown *pUnk);
        ~CHello();
        HRESULT __stdcall PrintSz(LPWSTR pwszString);
private:
        static int cObjRef;
};
#include "thlo.h"
#include "pshlo.h"
#include "shlo.hxx"
#include "mycls.hxx"
int CHello:cObjRef = 0;
CHello::CHello(IUnknown *pUnk) : CHelloBase(pUnk)
{
    cObjRef++;
    return;
}
HRESULT __stdcall CHello::PrintSz(LPWSTR pwszString)
{
    printf("%ws\n", pwszString);
    return(ResultFromScode(S_OK));
}
CHello::~CHello(void)
{
    cObjRef--;
    if( cObjRef == 0 )
        PulseEvent(hEvent);
    return;
}
#include "pshlo.h"
#include "shlo.hxx"
#include "mycls.hxx"
HANDLE hEvent;
int _cdecl main(int argc, char * argv[]) {
    ULONG ulRef;
    DWORD dwRegistration;
    CHelloCF *pCF = new CHelloCF();
    hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
    CoInitiali, NULL);
    CoInitializeEx(NULL, COINIT_MULTITHREADED);
    CoRegisterClassObject(CLSID_CHello, pCF, CLSCTX_LOCAL_SERVER,
        REGCLS_MULTIPLEUSE, &dwRegistration);
    WaitForSingleObject(hEvent, INFINITE);
    CoRevokeClassObject(dwRegistration);
    ulRef = pCF->Release();
    CoUninitialize();
    return(0);
}
extern CLSID CLSID_CHello;
extern UUID LIBID_CHelloLib;
CLSID CLSID_CHello = { 0x2573F891, 0xCFEE, 0x101A, { 0x9A, 0x9F, 0x00, 0xAA, 0x00, 0x34, 0x28, 0x20 } };
UUID LIBID_CHelloLib = { 0x2573F890, 0xCFEE, 0x101A, { 0x9A, 0x9F, 0x00, 0xAA, 0x00, 0x34, 0x28, 0x20 } };
#include "pshlo.h"
#include "shlo.hxx"
#include "clsid.h"
int _cdecl main( int argc, char * argv[]) {
    HRESULT hRslt;
    IHello *pHello;
    ULONG ulCnt;
    IMoniker * pmk;
    WCHAR wcsT[_MAX_PATH];
    WCHAR wcsPath[2 * _MAX_PATH];
    wcsPath[0] = '\0';
    wcsT[0] = '\0';
    if( argc 1) {
        mbstowcs(wcsPath, argv[1], strlen(argv[1]) + 1);
        wcsupr(wcsPath);
    }
    else {
        fprintf(stderr, "Object path must be specified\n");
        return(1);
    }
    if(argc 2)
        mbstowcs(wcsT, argv[2], strlen(argv[2]) + 1);
    else
        wcscpy(wcsT, L"Hello World");
    printf("Linking to object %ws\n", wcsPath);
    printf("Text String %ws\n", wcsT);
    hRslt = CoInitializeEx(NULL, COINIT_MULTITHREADED);
    if(SUCCEEDED(hRslt)) {
        hRslt = CreateFileMoniker(wcsPath, &pmk);
        if(SUCCEEDED(hRslt))
            hRslt = BindMoniker(pmk, 0, IID_IHello, (void **)&pHello);
        if(SUCCEEDED(hRslt)) {
            pHello->PrintSz(wcsT);
            Sleep(2000);
            ulCnt = pHello->Release();
        }
        else
            printf("Failure to connect, status: %lx", hRslt);
```

```
        CoUninitialize();
    }
    return(0);
}
```

MSIL

```
.assembly HelloWorld
{
}

.method static void Main()
{
    .entrypoint
    ldstr "Hello World!"
    call  void [mscorlib]System.Console::WriteLine(string)
    ret
}
```

GML

На **GML**:

```
show_message("Hello, World!");
```

или же

```
draw_text(42, 42, "Hello, World!");
```

можно и

```
show_error("Hello, World!", false);
```

Blitz3D, язык для новичков

Способ 1:

```
Text 0,0,"Hello, world!"
```

Способ 2:

```
Print "Hello, world!"
WaitKey ;Или Delay 5000
```

Способ 3, с выебонами:

```
Const HWtxt$ = "Hello, world!"

Graphics 800,600,32,2
SetBuffer BackBuffer()

 Type TCharacter
  Field c$
 End Type

GenerateHelloWorld

While Not KeyDown(1)
 Cls
 DrawHelloWorld
 Flip
Wend

End

Function GenerateHelloWorld()
 For i = 1 To Len(HWtxt)
  Local char.TCharacter = New TCharacter
   char\c = Mid(HWtxt,i,1)
 Next
End Function

Function DrawHelloWorld()
 For char.TCharacter = Each TCharacter
   i = i + 1
  Text i*10,10,char\c
 Next
End Function
```

Универсальный вариант

Универсальный Hello World! на C, C++, Haskell(нихуя, ghc это не ест), Ruby, Python, Perl(x2), HTML, tcl, zsh, make, bash и brainfuck

```
# /* [  <!-- */ include  <stdio.h> /*   \
  #{\
`""""true \\#{"\n#"};           \
  \
if [ -n "$ZSH_VERSION" ]; then           \
  \
  echo exec  echo I\'m a zsh script.; \
  \
elif [ -n "$BASH_VERSION" ]; then        \
  \
  echo exec  echo I\'m a bash script.; \
else  \
  echo exec  echo  I\'m  a sh  script.;    \
fi`;  #\
BEGIN{print"I'm a ", 0 ? "Ruby"  :"Perl",  " program.\n";  exit; }
  #\
%q~

set dummy =0; puts [list "I'm"  "a"  "tcl"  "script."]; exit

all: ; @echo "I'm a Makefile."          \
  #*/
/*: */ enum {a, b};            \
  \
static int c99(void) {

#ifndef __cplusplus /* bah */

unused1: if ((enum {b, a})0)          \
  (void)0;
#endif

unused2:  return a;     \
}  \
static int trigraphs(void) {         \
  \
  return sizeof  "??!"  ==  2;    \
}  \
char X;               \
  \
```

```
int main(void) {             \
  \
  struct X  {        \
  \
     char  a[2];    \
  };\
  if (sizeof(X)  !=  1) {       \
  \
printf("I'm a C++ program (trigraphs %sabled).\n",         \
  \
   trigraphs()  ? "en"  : "dis");\
  \
}else if (1//**/2


)unused3 : { ; \
    printf("I'm a C program (C%s, trigraphs %sabled).\n", \
        c99() ? "89 with // comments" : "99", \
        trigraphs() ? "en" : "dis"); \
  } else { \
    printf("I'm a C program (C89, trigraphs %sabled).\n", \
        trigraphs() ? "en" : "dis"); \
  } \
  return 0; \
} /*
# \
> main :: IO () -- -- \
> main = putStr "I'm a Literate Haskell program.\n"
# \
]>++++++++[<++++++++++>-]<+.>>++++[<++++++++++>-]<-.[-]>++++++++++ \
[<++++++++++>-]<-.>>++++[<++++++++>-]<.>>++++++++++[<++++++++++> \
-]<- - -.<.>+.->>++++++++++[<++++++++++>-]<++++.<.>>>++++++++++[ \
<++++++++++>-]<+++++.<<<<+.->>>>- - -.<+++.- - -<++.- ->>>>>++++ \
+++++[<++++++++++>-]<- - -.<<<<<.<+++.>>>.<<<-.- ->>>>+.<.<.<<.> \
++++++++++++++.[-]++++++++++"""`
# \
print "I'm a Python program."; """[-][--><html><head>
<!--:--><title>I'm a HTML page</title></head><body>
<!--:--><h1>I'm a <marquee><blink>horrible HTML</blink></marquee> page</h1>
<!--:--><script language="Javascript">
<!--: # \
setTimeout( // \
  function () { // \
   document.body.innerHTML = "<h1>I'm a javascript-generated HTML page</h1>"; // \
  }, 10000); // \
//-->
</script><!--: \
</body></html><!-- } # \
say "I'm a Perl6 program", try { " ($?PUGS_VERSION)" } // "", "."; # """ # */
#define FOO ]-->~
```

NATURAL

```
define data local
1 #BUTTON(A1)
end-define
Write notitle 'Hello, world!'
display notitle 'Hello, world!'
print 'Hello, world!'
PROCESS GUI ACTION MESSAGE-BOX WITH NULL-HANDLE
  'Hello, world!' ' ' 'IO1'
  #BUTTON
  GIVING *ERROR-nr
Input (ad=od) 'Hello, world!'
end
```

LSL

```
default
{
    state_entry()
    {
      llSay(0,"Hello, world!");
    }
}
```

Erlang

```
#!/usr/bin/env escript

main(_Args) ->
        io:format("Hello, World!~n").
```

Malbolge

```
(=<`:9876Z4321UT.-Q+*)M'&%$H"!~}|Bzy?=|{z]KwZY44Eq0/{mlk**hKs_dG5[m_BA{?-Y;;Vb'rR5431M}/.zHGwEDCBA@98\6543W10/.R,+O<
```

LOLCODE

```
HAI
CAN HAS STDIO?
VISIBLE "HAI WORLD!"
KTHXBYE
```

Go

```
package main

import "fmt"

func main() {
        fmt.Println("Hello, world!")
}
```

или проще:

```
package main

func main() {
        println("Hello, world!")
}
```

ProceRaptor

```
application ($console,$void);
  var ("st",$string);
  "Hello World!">>st;
  write (st,$line);
  read ($void,$line);
exit();
```

Icon

```
procedure main()
write("Hello, world!")
end
```

ABAP

```
REPORT  zhello.
WRITE / 'Hello, World!'.
```

или так

```
REPORT  zhello.
MESSAGE 'Hello, World!' TYPE 'I'.
```

или так

```
REPORT  zhello.

CALL FUNCTION 'POPUP_TO_DISPLAY_TEXT'
  EXPORTING
    titel        = 'Helloworld'
    textline1    = 'Hello'
    textline2    = 'World!'
    start_column = 25
    start_row    = 6.
```

Befunge

```
> # # # # # # # # # 0 v
> #,_@#:"Hello, world!"
```

или так

```
0"!dlroW ,olleH">:#,_@
```

TeX

```
\documentclass[a4paper,12pt]{article}
\usepackage[T2A]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}
\usepackage[pdftex,unicode]{hyperref}
\begin{document}
Hello, World!
\end{document}
```

(на самом деле предыдущий оратор показал код на LaTeX, а на православном TeX это выглядит куда проще):

```
Hello, World!
```

Не совсем так:

```
Hello, World!
\bye
```

Vimscript

```
:echo 'Hello World!'
```

Small Basic

```
TextWindow.WriteLine("Hello, World!")
```

Rust

```
fn main() {

    println!("Hello World!");
}
```

Swift 2.1

```
print("Hello, world!")
//либо так
let hello = "Hello World!"
print(hello)
```

MATLAB

```
disp('Hello, world!');
```

ArnoldC

```
IT'S SHOWTIME
TALK TO THE HAND "hello world"
YOU HAVE BEEN TERMINATED
```

V

```
_Счетчик Отобразить(_Текст) импорт _ОтобразитьТекст;
->ПриветМир():
    Отобразить "Hello, world!";
    <-0;
.
```

ЯСОП52

```
 /*@@ПриветЧел
 /* запрет вывода на чек долбанных номеров операций
 ВИД_ПЕЧАТИ(Пнс=1);
 /* привлечение внимания к оборудованию
 ВЫЗОВ(ПП=МерзкаяПищалка,прм1=3);
 /* запрос фамилии страдальца
 ВВОД_КЛ(ТИПП=С,ТЕКСТ="РУЧНАЯ ОПЕРАЦИЯ 666:ВВЕДИТЕ ФАМИЛИЮ     ОПЕРАТОРА",ИМЯ=фамОпер);
 /* вывод на чек и на экран
 ВЫВОД(ВД="ПРИВЕТ,",ПОЗ=1,ПЕЧАТЬ=ДА);
 ВЫВОД(ВД=фамОпер,ПОЗ=21,ПЕЧАТЬ=ДА);
 /* задержка времени 3.6 сек, чтобы оператор успел прочитать на экране сообщение и уронить слезу
 ЗАДЕР_В_МС(Т=3600);
 ФИНИШ;
 /*
 /*==========
 /* <hate> мерзко пищит,
```

```
/* использовать против людей
/*==========
МерзкаяПищалка: НАЧАЛО(прм1=ЦБ2);
/* прм1 - количество повторов звукового сигнала
ПРИСВОИТЬ(ТИПП=ЦБ2,ЗНАЧ=прм1,ИМЯ=ошПерТумбл4);
НЦ(НАЧ=1,КОН=ошПерТумбл4,ШАГ=1,ИНД=И);
 ЗВУК(Пс=ВКЛ);
 ЗАДЕР_В_МС(Т=800);
 ЗВУК(Пс=ОТКЛ);
 ЗАДЕР_В_МС(Т=1200);
КЦ;
КОНЕЦ;
/*==========
/*
```