

Catalyde: Un entorno de iniciación a la programación con fines educativos

Catalin Costin Stanciu

Universitat Politècnica de València

25 de febrero de 2013



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escuela Técnica
Superior de Ingeniería
Informática

- 1 Introducción
- 2 Demostración
- 3 Descripción de la aplicación
- 4 Tecnologías
- 5 Funcionamiento del sistema de compilación
- 6 Conclusiones y trabajo futuro

Presentación del problema

- Programar supone ser capaz de convertir un modelo mental de "*lo que tiene que hacer o resolver un programa*" en el diseño y la posterior implementación que lo realice.
- Los alumnos deben centrarse en la parte algorítmica y no distraerse con detalles.
- Entender lo que se pide en un ejercicio de programación es esencial y muchas veces los alumnos no entienden realmente lo que el profesor pide con un ejercicio.
- Doble esfuerzo al iniciarse en el mundo de la programación:
 - Aprender a programar.
 - Aprender a usar el entorno.

Presentación del problema

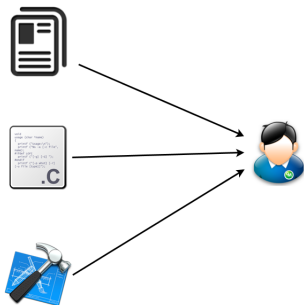
- Programar supone ser capaz de convertir un modelo mental de "*lo que tiene que hacer o resolver un programa*" en el diseño y la posterior implementación que lo realice.
- Los alumnos deben centrarse en la parte algorítmica y no distraerse con detalles.
- Entender lo que se pide en un ejercicio de programación es esencial y muchas veces los alumnos no entienden realmente lo que el profesor pide con un ejercicio.
- Doble esfuerzo al iniciarse en el mundo de la programación:
 - Aprender a programar.
 - Aprender a usar el entorno.

Presentación del problema

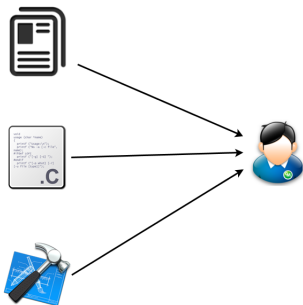
- Programar supone ser capaz de convertir un modelo mental de "*lo que tiene que hacer o resolver un programa*" en el diseño y la posterior implementación que lo realice.
- Los alumnos deben centrarse en la parte algorítmica y no distraerse con detalles.
- Entender lo que se pide en un ejercicio de programación es esencial y muchas veces los alumnos no entienden realmente lo que el profesor pide con un ejercicio.
- Doble esfuerzo al iniciarse en el mundo de la programación:
 - Aprender a programar.
 - Aprender a usar el entorno.

Presentación del problema

- Programar supone ser capaz de convertir un modelo mental de "*lo que tiene que hacer o resolver un programa*" en el diseño y la posterior implementación que lo realice.
- Los alumnos deben centrarse en la parte algorítmica y no distraerse con detalles.
- Entender lo que se pide en un ejercicio de programación es esencial y muchas veces los alumnos no entienden realmente lo que el profesor pide con un ejercicio.
- Doble esfuerzo al iniciarse en el mundo de la programación:
 - Aprender a programar.
 - Aprender a usar el entorno.



- El **profesor** proporciona al alumno una serie de contenidos:
 - Enunciado
 - Código para rellenar (opcional)
- El **alumno**:
 - Utiliza un entorno de desarrollo (más o menos complejo) para escribir y depurar el programa.
 - No se puede comprobar el funcionamiento hasta que se haya completado todo el programa.



- El **profesor** proporciona al alumno una serie de contenidos:
 - Enunciado
 - Código para rellenar (opcional)
- El **alumno**:
 - Utiliza un entorno de desarrollo (más o menos complejo) para escribir y depurar el programa.
 - No se puede comprobar el funcionamiento hasta que se haya completado todo el programa.

¿Qué son?

- Gran auge de los cursos a distancia debido a las plataformas *online*.
- Aparición de los Cursos Abiertos En línea Masivos (en inglés Massive Open Online Course(MOOC)).
- Existen varias plataformas (y cada vez más). Muchas relacionadas con la computación y, en concreto, con la programación.

Cursos Abiertos Masivos En Línea

Variables and Data Types

Curso creado por [Eric Weinberg](#)

1. Welcome to the Flying Circus

Python is a powerful, flexible programming language you can use to aid in development, to write desktop graphical user interfaces (GUIs), create games, and much more. Python is:

- "High-level", meaning reading and writing Python is really easy—it looks a lot like regular English.
- "Interpreted", meaning you don't need a compiler to write and run Python. You can write it here at Coursera.org or even on your own computer (many are adapted with the Python Interpreter built-in—wait until the interpreter later in this lesson).

• "Object-oriented", meaning it allows users to manipulate data structures called objects in order to build and execute programs. We'll learn more about objects later.

• Fast to use. Python is named after Monty Python's Flying Circus, and example code and tutorials often refer to the show and include jokes in order to make learning the language more interesting.

This course assumes no previous knowledge of

UDACITY CS215

Overview **Classroom** Discussion Wiki Announcements Progress

Up Heapify

```
1 # write up_heapify, an algorithm that checks if
2 # node i and its parent satisfy the heap
3 # property, swapping and reheapifying if they don't
4 # i should be a heap when up_heapify is done
5
```

```
6 def up_heapify(it, i):
7     if i == 0:
8         return
9     p = i // 2
10    if it[i] < it[p]:
11        it[i], it[p] = it[p], it[i]
12        up_heapify(it, p)
13    return
14
15 def parent(i):
16     return (i-1)//2
17
18 def test_up_heapify():
19     return 2*1+1
```

Run to see results

Run to see any potential errors

Instructor Comments

Supplementary Material

Sección **Cursos de preguntas y respuestas (31)** [Glosario](#) [Búsqueda avanzada](#)



My Classes - [Welcome, Joan Pastor](#)



Cursos Universidades **MI** Página Blog

Android: Programación de Aplicaciones

[Inicio](#) [PyR](#) [Foro](#) [Blog](#)

Módulo 1: Visión general y entorno de desarrollo

Información

Contenidos

- ✓ ¿Qué hace Android exactamente?
- ✓ ¿Por qué hacer Android exactamente?
- ✓ Comparativa con otras plataformas
- ✓ Test. Compatibilidad con otras plataformas
- ✓ Test. Compatibilidad con otras plataformas
- ✓ Arquitectura de Android
- Test. Arquitectura de Android
- Instalación del entorno de desarrollo
- Creación de un dispositivo virtual
- Android (AVD)
- Test. - Instalación
- Los versiones de Android y niveles de API
- Test. - Plataformas de desarrollo
- Creación de un programa
- Ejecución del programa

Introducción

La telefonía móvil está cambiando la sociedad actual de una forma tan significativa como lo ha hecho Internet. Esta revolución no ha hecho más que empezar, los nuevos terminales ofrecen unas capacidades similares a un ordenador personal, lo que permite que puedan ser utilizados para leer nuestro correo o navegar por Internet. Pero a diferencia de un ordenador, un teléfono móvil siempre está en el bolsillo del usuario. Esto permite un nuevo abanico de aplicaciones mucho más cercanas al usuario. De hecho, muchos autores coinciden en que el nuevo ordenador personal del siglo veintiuno será un terminal móvil.

El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado una gran expectación y está teniendo una importante aceptación tanto por los usuarios como por la industria. En la actualidad se está convirtiendo en una seria alternativa frente a otras plataformas como Symbian, iPhone o Windows Phone.

A lo largo de esta unidad veremos también cómo instalar y trabajar con el entorno de desarrollo (Eclipse + Android SDK), además de describir la estructura de un proyecto Android.



Buscar en Internet

3 resultados
16 de enero de 2013



Valoración de Futbolistas

3 resultados
16 de enero de 2013

Herramienta propuesta

- Catalyde: Catalyzer Development Environment.
- La aplicación Catalyde introduce a los usuarios (alumnos) en la disciplina de la programación.
- La herramienta está destinada a estudiantes de primeros cursos.
- Aporta ventajas al alumno y al profesor.



Herramienta propuesta

- Catalyde: Catalyzer Development Environment.
- La aplicación Catalyde introduce a los usuarios (alumnos) en la disciplina de la programación.
- La herramienta está destinada a estudiantes de primeros cursos.
- Aporta ventajas al alumno y al profesor.



Herramienta propuesta

- Catalyde: Catalyzer Development Environment.
- La aplicación Catalyde introduce a los usuarios (alumnos) en la disciplina de la programación.
- La herramienta está destinada a estudiantes de primeros cursos.
- Aporta ventajas al alumno y al profesor.



Herramienta propuesta

- Catalyde: Catalyzer Development Environment.
- La aplicación Catalyde introduce a los usuarios (alumnos) en la disciplina de la programación.
- La herramienta está destinada a estudiantes de primeros cursos.
- Aporta ventajas al alumno y al profesor.



- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

Aportaciones de la aplicación

- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

Aportaciones de la aplicación

- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

Aportaciones de la aplicación

- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

Aportaciones de la aplicación

- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

The screenshot shows the Catalyde IDE interface. The top menu bar includes 'Catalyde', 'Archivos', 'Opciones', 'Catalin Stanciu', 'Acercas de', and 'Salir'. On the left, a sidebar titled 'Prácticas' lists 'Práctica 1', 'Práctica 0', 'Práctica 4', and 'Práctica 8' (which is selected and highlighted in blue). Below this is an 'Índice' link. The main editor area is titled 'Práctica 8' and contains the following C code:

```
int main() {  
    char cadena[TAM];  
    printf("Escribe lo que te de la gana: ");  
    fgets(cadena, TAM, stdin); /* lee hasta fin de línea */  
    printf("Lo que has escrito tiene %d vocales,"  
          " no te da vergüenza?\n", contar_vocales(cadena));  
    return 0;  
}
```

Below the code, there is a paragraph of text in Spanish: "Para realizar la función contar_vocales debes hacer primero la siguiente función que recibe un carácter y devuelve 0 si NO es vocal y 1 si lo es, usar un switch para ver si el caracter en cuestión es una vocal:"

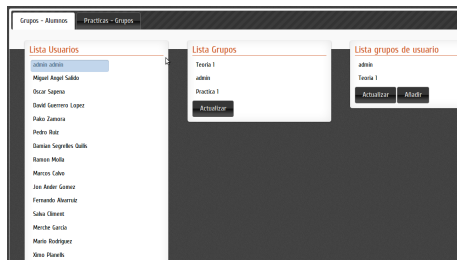
Below the text, there is another code block for the `es_vocal` function:

```
int es_vocal(char caracter) {  
  
    switch(caracter)  
    {  
        case 'a': |  
        case 'e': |  
        case 'i': |  
        case 'o': |  
        case 'u': |  
        return 1;  
        default:  
        return 0;  
    }  
}
```

Below the code block, there is a toolbar with buttons: 'Ejecutar', 'Deshacer', 'Rehacer', and 'Reindentar'. The 'Código activado' button is also visible.

Vista del profesor

- Sistema de edición de prácticas.
- Interfaz de administración.



```
1 ## Ejercicio sencillo para completar
2
3 Completa el siguiente programa que calcula el área de un rectángulo. Para ello debes:
4
5 - Escribir una función llamada 'area_rectangulo' para que se pueda utilizar en el programa
  la función).
6 - Completar el programa principal para que pida la altura.
7
8 ...
9 #include <stdio.h>
10 ...
11
12 <!-- BEGINBLOCK { "b_tag": "funcion_area_rectangulo", "type": "code", "checkbox": "true", "edit":
  [{"name": "Ejecutar", "action": "run_area"}]} -->
13 /* aquí va el código de la función area_rectangulo */
14 <!-- ENDBLOCK -->
15
16 ...
17 int main() {
18     float base, altura, area;
19     printf(stdout, "Introduce la base: ");
20     scanf(stdin, "%f", &base);
21     ...
22
23 <!-- BEGINBLOCK { "b_tag": "pedir_altura", "type": "code", "checkbox": "true", "edit":
  [{"name": "Ejecutar", "action": "run_area"}]} -->
24 /* aquí va el código que pide la altura */
25 <!-- ENDBLOCK -->
26
27 ...
28     area = area_rectangulo(base, altura);
29     printf(stdout, "El área es: %f\n", area);
30     return 0;
31 }
32 ...
33
```

Cliente

- El cliente se encarga de:
 - Interfaz de usuario.
 - Interacción con el programa ejecutado mediante la terminal.

Servidor

- El servidor se encarga de:
 - Autenticación de usuarios.
 - Gestión de grupos y prácticas.
 - Servir una práctica: mostrar el boletín con los bloques a editar, los botones de configuración y de ejecutar, etc.
 - Realizar las acciones de compilar y ejecutar.

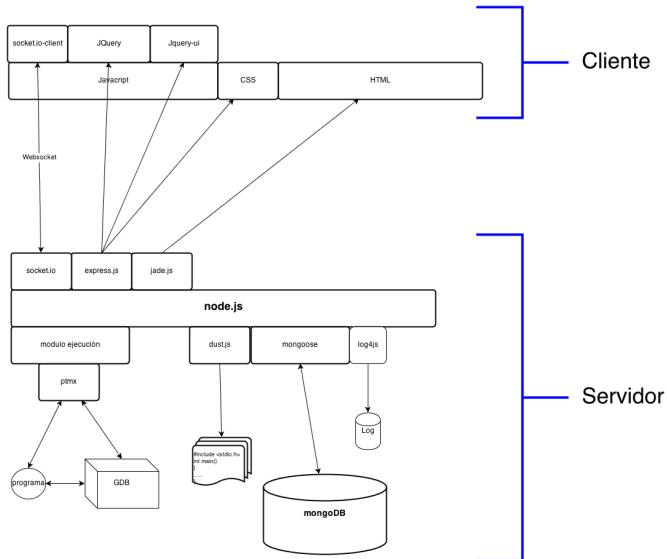
Cliente

- El cliente se encarga de:
 - Interfaz de usuario.
 - Interacción con el programa ejecutado mediante la terminal.

Servidor

- El servidor se encarga de:
 - Autenticación de usuarios.
 - Gestión de grupos y prácticas.
 - Servir una práctica: mostrar el boletín con los bloques a editar, los botones de configuración y de ejecutar, etc.
 - Realizar las acciones de compilar y ejecutar.

Arquitectura





Nodejs

- Librería asíncrona dirigida por eventos.
- Máquina virtual Javascript V8.



Express

- Convierte nodejs en un servidor web proporcionado:
 - Permisos de acceso
 - Rutas
 - Servidor contenido estático

Mongodb

- Base de datos NoSQL
- Rápido acceso a los datos y fácilmente escalable

Nodejs

- Librería asíncrona dirigida por eventos.
- Máquina virtual Javascript V8.



Express

- Convierte nodejs en un servidor web proporcionado:
 - Permisos de acceso
 - Rutas
 - Servidor contenido estático

Mongodb

- Base de datos NoSQL
- Rápido acceso a los datos y fácilmente escalable

Nodejs

- Librería asíncrona dirigida por eventos.
- Máquina virtual Javascript V8.



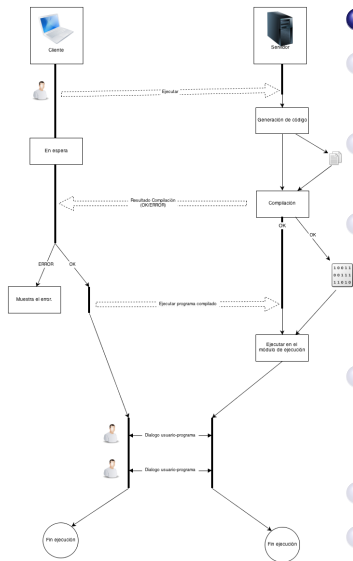
Express

- Convierte nodejs en un servidor web proporcionado:
 - Permisos de acceso
 - Rutas
 - Servidor contenido estático

Mongodb

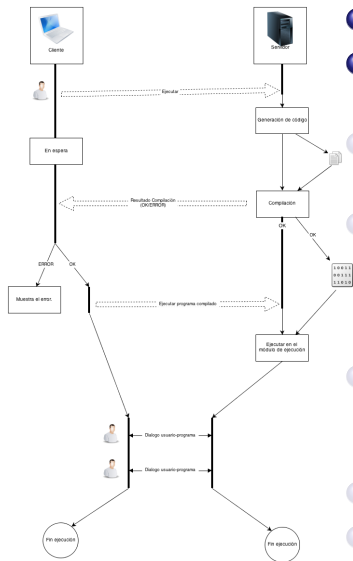
- Base de datos NoSQL
- Rápido acceso a los datos y fácilmente escalable

Compilación y ejecución



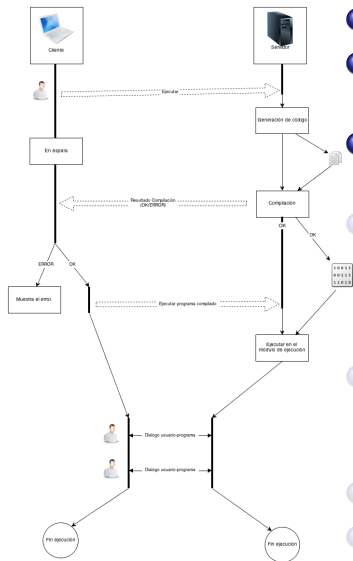
- 1 El cliente manda el comando de ejecutar.
- 2 El servidor construye los ficheros fuente y pasa a la fase de compilación.
- 3 El servidor compila el código y devuelve la respuesta al cliente.
- 4 El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
- 5 Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
- 6 El cliente interacciona con el programa.
- 7 Se finaliza la ejecución.

Compilación y ejecución



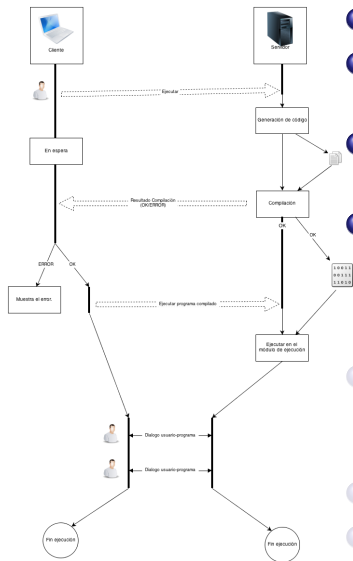
- 1 El cliente manda el comando de ejecutar.
- 2 El servidor construye los ficheros fuente y pasa a la fase de compilación.
- 3 El servidor compila el código y devuelve la respuesta al cliente.
- 4 El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
- 5 Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
- 6 El cliente interacciona con el programa.
- 7 Se finaliza la ejecución.

Compilación y ejecución



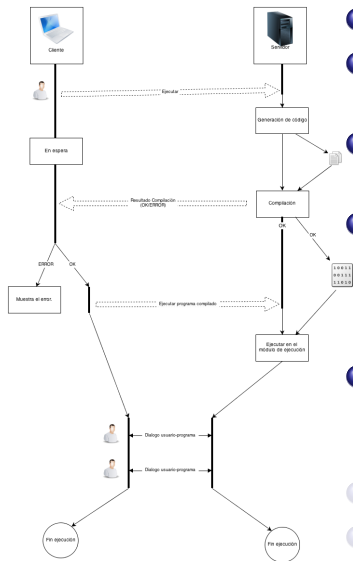
- 1 El cliente manda el comando de ejecutar.
- 2 El servidor construye los ficheros fuente y pasa a la fase de compilación.
- 3 El servidor compila el código y devuelve la respuesta al cliente.
- 4 El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
- 5 Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
- 6 El cliente interacciona con el programa.
- 7 Se finaliza la ejecución.

Compilación y ejecución



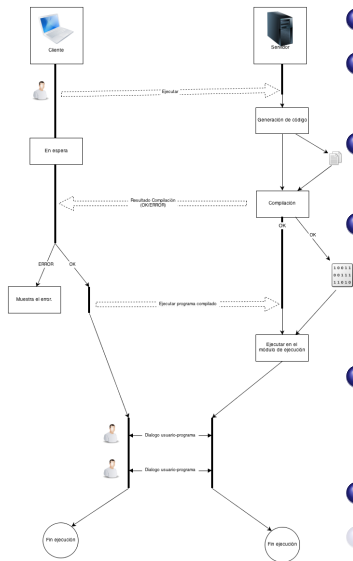
- 1 El cliente manda el comando de ejecutar.
- 2 El servidor construye los ficheros fuente y pasa a la fase de compilación.
- 3 El servidor compila el código y devuelve la respuesta al cliente.
- 4 El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
- 5 Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
- 6 El cliente interacciona con el programa.
- 7 Se finaliza la ejecución.

Compilación y ejecución



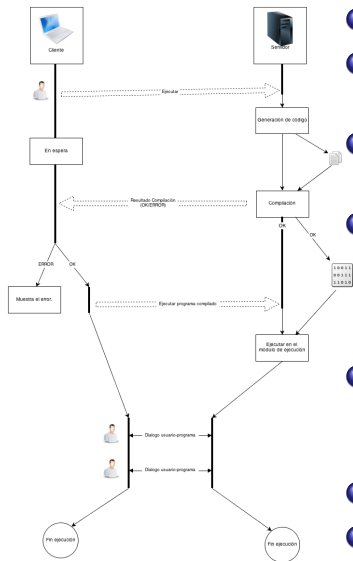
- 1 El cliente manda el comando de ejecutar.
- 2 El servidor construye los ficheros fuente y pasa a la fase de compilación.
- 3 El servidor compila el código y devuelve la respuesta al cliente.
- 4 El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
- 5 Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
- 6 El cliente interactúa con el programa.
- 7 Se finaliza la ejecución.

Compilación y ejecución



- 1 El cliente manda el comando de ejecutar.
- 2 El servidor construye los ficheros fuente y pasa a la fase de compilación.
- 3 El servidor compila el código y devuelve la respuesta al cliente.
- 4 El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
- 5 Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
- 6 El cliente interacciona con el programa.
- 7 Se finaliza la ejecución.

Compilación y ejecución



- 1 El cliente manda el comando de ejecutar.
- 2 El servidor construye los ficheros fuente y pasa a la fase de compilación.
- 3 El servidor compila el código y devuelve la respuesta al cliente.
- 4 El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
- 5 Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
- 6 El cliente interacciona con el programa.
- 7 Se finaliza la ejecución.

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.



Catalin Costin Stanciu, Joan Pastor-Pellicer, and Salvador España Boquera. Catalyde: herramienta para enseñar programación. In V Jornada de Innovación Docente, Valencia, Spain, December 2012. URL: <http://jidinf12.webs.upv.es/poster>.

Gracias por su atención.

Catalyde Archivos Opciones Catalin Stanciu Acerca de Salir

Prácticas

Practica 1

Práctica 0

Práctica 4

Práctica 8

Índice

Práctica 4

Completa el siguiente programa que calcula el área de un rectángulo. Para ello debes:

- Escribir una función llamada a `area_rectangulo` para que se pueda utilizar en el programa principal (observa cómo se realiza la llamada a la función).
- Completar el programa principal para que pida la altura.

#include <stdio.h>

Código desactivado

Ejecutar

Deshacer

Rehacer

Reindentar

1 /* aqui va el codigo de la función area_rectangulo */

int main() {
float base, altura, area;
fprintf(stdout, "Introduce la base: ");
fscanf(stdin, "%f", &base);

Código desactivado

Ejecutar

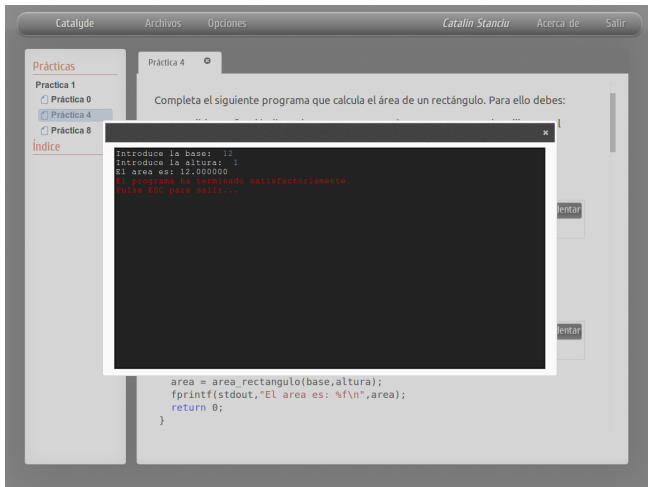
Deshacer

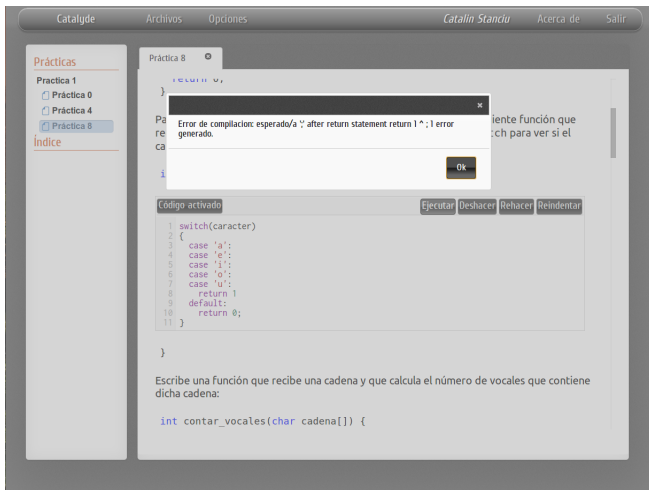
Rehacer

Reindentar

1 /* aqui va el codigo que pide la altura */

area = area_rectangulo(base, altura);
fprintf(stdout, "El area es: %f\n", area);
return 0;
}





Prácticas

Practica 1

Práctica 0

Práctica 4

Práctica 8

indice

Guarda actual

Guardarlo todo

```
int main() {
    char cadena[TAM];
    printf("Escribe lo que te de la gana: ");
    fgets(cadena, TAM, stdin); /* lee hasta fin de linea */
    printf("Lo que has escrito tiene %d vocales,"
           " no te da verguenza?\n", contar_vocales(cadena));
    return 0;
}
```

Para realizar la función `contar_vocales` debes hacer primero la siguiente función que recibe un carácter y devuelve 0 si NO es vocal y 1 si lo es, usar un `switch` para ver si el carácter en cuestión es una vocal:

```
int es_vocal(char character) {
```

Código activado

Ejecutar

Deshacer

er Rehacer

Reindentar

```
1 switch(caracter)
2 {
3     case 'a': |
4     case 'e':
5     case 'i':
6     case 'o':
7     case 'u':
8         return 1;
9     default:
10        return 0;
11 }
```