

Haskell-IOS

Aplicación demo para ejecutar código Haskell desde dispositivos mobiles IOS

Integrantes

- Ernesto Manuel Carrión Chumpitaz
- Alejandro Aristizabal Londoño
- Daniel Munera Sanchez

Profesor

- Francisco Jose Correa Zabala

Introducción

Este documento describe la instalación y el uso del GHC (Glasgow Haskell Compiler) en dispositivos IOS

Instalación

- Descargar e instalar [GHC standar](#)
- Extraer en el directorio raíz “/” el archivo [ghc-iphone.tar.bz](#)

Haskell Backend

Main.hs

```
{-# LANGUAGE ForeignFunctionInterface, EmptyDataDecls #-}

import Foreign.C.Types

foreign import ccall safe "openWindow" openWindow
    :: IO CInt

foreign export ccall "fibonacci" fibonacci
    :: Int -> Int

foreign export ccall "fac" fac
    :: Int -> Int

fibonacci :: Int -> Int
fibonacci n = fibs !! n
    where fibs = 0 : 1 : zipWith (+) fibs (tail fibs)

fac :: Int -> Int
fac 0 = 1
fac n | n > 0 = n * fac (n-1)

main = do
    openWindow
    return ()
```

Haskel_IOS.cabal

```
Name: HaskellIOS
Version: 0.1
Synopsis: iPhone app in haskell
Description: iPhone app in haskell
Build-type: Simple
Cabal-Version: >= 1.6

Executable Haskell_IOS
    Main-Is: Main.hs
    Frameworks:
        Foundation
        UIKit
        CoreGraphics
    Build-Depends:
        base >= 4
```

Configuración XCode

En el dispositivo - Desabilitar ‘Thumb Mode’

- Abrir las propiedades del proyecto
- En el Target del proyecto seleccionar el tab **Build Settings**
- Buscar por **thumb** y deseleccionar **Compile for thumb**

Añadir Haskell Target

El proyecto de Xcode debe construir automaticamente el proyecto Haskell, previamente se deben haber agregado a la raíz del proyecto el archivo *.hs y el archivo *.cabal

- Seleccionar File > New > New Target
- Seleccionar **Other** dentro del grupo **Mac OSX**
- Seleccionar **External Build System**
- Escribir **Haskell** como Product Name (o cualquier normbre de su preferencia)
- Escribir **/opt/iphone/bin/build-iphone-haskell.s** en el campo **Build Tool**
- Establecer el Target **Haskell** como dependencia del proyecto:
 - Doble click en el Target principal del proyecto
 - Seleccionar el tab **Build Phases**
 - Expandir la lista **Target Dependencies** y arrastrar el Targer **Haskell** hacia allí

Construir Haskell Target

Necesitamos compilar primero el Target **Haskell** para poder generar una librería con los fuentes de haskell para poder despues añadirla al proyecto.

- Seleccionar Product > Build al Target **Haskell**
- Doble click en el panel de errores y revisar que no haya ningún error en los fuentes de haskell. Correguir los errores hasta que no quede ningún error.
- Es pobable que la compilación del proyecto principal falle, pero lo importante es que el Target **Haskell** compile correctamente.

Añadir libreria al proyecto

Esto nos va a permitir ejecutar código haskell desde un FrontEnd **Objective-C** mediante un wrapper en **C**

- Doble click en el Target principal del proyecto
- Seleccionar el tab **Build Phases**
- Expandir la lista **Link Binary With Libraries**
- Añadir la librería generada anteriormente en el path definido por Xcode donde se guardan todos los Builds **Build Dir**

Haskell desde Objective-C

main.m

```
#import <UIKit/UIKit.h>

extern int Haskell_main(int argc, char* argv[]);

int main(int argc, char *argv[])
{
    Haskell_main(argc, argv);
}

int doMain(int argc, char *argv[]) {

    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    int retVal = UIApplicationMain(argc, argv, nil, nil);
    [pool release];
    return retVal;
}

int openWindow(void);

int openWindow() {
    static char* args[2];
    args[0] = "dummy";
    args[1] = NULL;
    return doMain(1, args);
}
```

HaskellViewController.m

```
extern int fibonacci(int a1);
extern int fac(int n);
```

HaskellViewController.m

```
- (IBAction)fibPressed {

    int i = [self.input_.text intValue];
    int o = fibonacci(i);
    self.output_.text = [NSString stringWithFormat:@"%d", o];
}

- (IBAction)facPressed {

    int i = [self.input_.text intValue];
    int o = fac(i);
    self.output_.text = [NSString stringWithFormat:@"%d", o];
}
```

Conclusiones

- Lograr la ejecución de código **Haskell** desde **Objective-C** nos ha brindado una verdadera integración de paradigmas en una sola plataforma:
 - Haskell: Funcional
 - Objective-C: Orientado a Objetos
 - C: Imperativo/Estructurado
- Gracias a la ejecución de código Haskell desde Objective-C hemos podido ampliar nuestro espectro de soluciones en la plataforma IOS, como también podemos mejorar ciertas tareas que tengan una naturaleza funcional, tales como:
 - Inteligencia Artificial
 - Diseño de lenguajes de alto nivel

TODO

- Ya que el Glasgow Haskell Compiler (GHC) se encarga de convertir código Haskell en **C** debe ser posible poder usar esta librería para ejecutar código Haskell desde dispositivos android usando el NDK (Native development Kit)
- Al contar Android con una **Scripting layer** es posible ejecutar **código Python** lo cual nos podría permitir usar sus las características funcionales de ese lenguaje
- Investigar sobre posibilidades de ejecutar programas funcionales en los demás sistemas operativos de los smartphones restantes:
 - BlackBerry OS - RIM
 - Windows Phone 7 - Microsoft
 - Symbian - Nokia
 - Meego - Nokia/Intel

Referencias

- [Developing iphone applications in Haskell](#)
- [Iphone Port of GHC](#)
- [A Haskell to Objective-C Binding](#)
- [A small Haskell / Objective-C Interface](#)
- [Using Haskell in an XCode Cocoa Project](#)
- [Calling Haskell from C](#)
- [Haskell in Industry](#)