**Instituto Tecnológico de Costa Rica**

I tAREA pROGRAMADA (pROGRAMACIÓN Imperativa)

Sede San Carlos 18/04/2017

## Unidad de Computación

# **Jose Antonio Herrera Ramírez**

# Introducción

Hoy en día la mayoría de los proyectos software se desarrollan dentro del paradigma imperativo. Este paradigma se caracteriza por ser dinámico, ya que el programa puede pasar por distintos estados durante la ejecución del mismo y se puede alterar fácilmente su comportamiento, o bien los espacios de memoria que este procesa.

En este proyecto se realizará un programa para procesar instrucciones bytecode definidas en el alcance del proyecto, las cuales serán generadas en un archivo externo por el compilador de Python.

El lenguaje bajo el cual se desarrollará el proyecto es ‘C’, el cual permite un manejo complejo de los espacios de memoria o de los estados del programa.

# Análisis del problema

El problema consiste en identificar grupos de casillas presentes en un array bidimensional(matriz) mediante el uso de funciones predefinidas del lenguaje y también de funciones definidas por el desarrollador de manera recursiva.

Asimismo, el usuario final deberá poder interactuar con el programa mediante una interfaz gráfica, desde donde podrá acceder a las consultas desarrolladas y establecer la comunicación con el sistema.

Se deberá programar la lógica del programa de manera funcional para luego poder tener acceso e implementar las funciones desde el componente de interfaz.

# Solución del problema

Para la solución se desarrollaron dos “módulos”, el lógico, que contiene todas las funciones recursivas que serán usadas a lo largo del programa, y el módulo de interfaz para proveer la facilidad de comunicación usuario-sistema.

## Logica

En el módulo lógico, para obtener los distintos grupos en la matriz se implementaron un conjunto de funciones, dentro de las cuales resaltan las siguientes:

**Vecinos:** retorna los puntos(x,y) de las casillas ocupadas anexas a un punto dado.

**Miembro o Contenido:** se utiliza para verificar si un punto(x,y) dado se encuentra en una lista dada, esta función es útil para verificar si ya un punto está en un grupo.

**Extender:** recibe un punto (x,y) y retorna una lista con todos los vecinos de los vecinos, y así iterando de manera recursiva, al final el valor que retorna será el conjunto total de grupos, esta función va de la mano con otra función auxiliar que se encarga de armar la lista de cada uno de los grupos existentes.

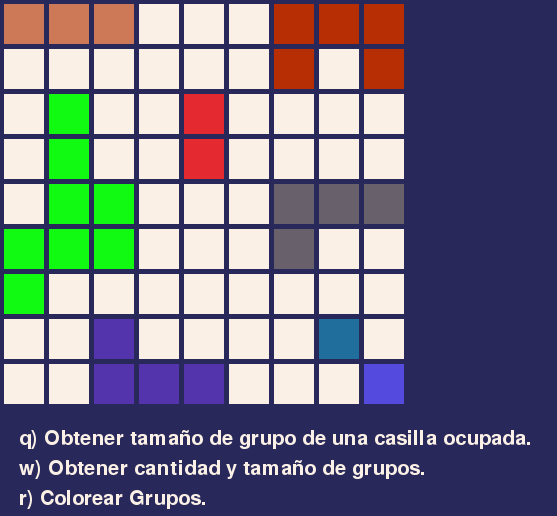
Las demás funciones pertenecen a los requerimientos y consultas del programa, por ejemplo, la función de llenar la matriz de manera aleatoria, buscar el tamaño del grupo de un punto dado, entre otras.

## Interfaz grafica

El módulo de interfaz se realizó con la ayuda de la librería Pygame, la cual provee una serie de métodos para manejo de interfaces de usuario, eventos, renderizado de geometrías y demás.

Esta interfaz estará ligada a la lógica implementada en otro archivo, por lo cual solo es necesario importar el archivo y usar las funciones del módulo lógico

Primeramente, se le dan al jugador las opciones de llenar la matriz o que el sistema genere una matriz aleatoria apoyándose en una función del módulo lógico. Luego el usuario selecciona el tamaño de la matriz y el sistema lo redirigirá a una pantalla como la que se muestra a continuación:

Desde acá el usuario podrá ver la lista de grupos gráficamente ubicados en la matriz y resaltados con distintos colores, además tendrá acceso a las consultas del menú donde podrá teclear la opción que guste.

# Análisis de los resultados

De las tareas o procesos desarrollados existen algunos que no están optimizados, por lo que la comprensión de este apartado es vital para una manipulación correcta del programa, caso contrario podrían darse errores en la ejecución.

En la siguiente tabla se desglosan las actividades o partes del sistema con su estado y observaciones.

|  |  |  |
| --- | --- | --- |
| **Tarea** | **Estado** | **Observaciones** |
| Funciones del módulo lógico | Incompleto | Falta la función de los grupos distintos con sus tamaños |
| Interfaz grafica | No optimizada | Únicamente teclear las opciones que aparezcan en la vista actual, de lo contrario se podrían dar imprevistos en el comportamiento del programa. Al entrar a la vista de la matriz no se puede devolver al menú principal, por lo que es necesario cerrar el programa y volverlo a abrir si quisiera utilizar otra opción del menú.  Los resultados de las consultas solo se podrán ver desde consola. |

# Conclusiones y Recomendaciones

Se logró determinar que un programa escrito en lenguaje funcional optimiza la reducción de código, aunque reduzca un poco la claridad del código al tratarse de varias funciones indentadas.

El lenguaje utilizado es una buena opción para adentrarse en la programación funcional debido a la familiaridad y facilidad que provee al tratarse de un lenguaje de muy alto nivel

Se recomienda el uso de motores o librerías de interfaz gráfica que provean funcionalidades más robustas

# Bibliografía

DeNero, J. (s.f.). *Composing Programs: Recursive Functions*. Obtenido de Composing Programs: http://composingprograms.com/pages/17-recursive-functions.html

Foundation, T. P. (s.f.). *Python Docs: Chapter 5.Data Structures*. Obtenido de Python Docs: https://docs.python.org/2/tutorial/datastructures.html

Mertz, D. (2015). *Functional Programming in Python.* O'Reilly.

.