DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Semana 5:
Tipos de Datos Abstractos

Semestre 2 - 2024

Análisis de Algoritmos y Estructura de Datos

Departamento de Ingeniería Informática

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Contenido

- Especificación de TDA
- Implementación en *C* de TDA

Objetivos

- Implementar tipos de datos abstractos utilizando el lenguaje de programación *C*.

Actividad 1: TDA punto

- Sea P un punto en el plano cartesiano, especificar el dominio (estructura de datos) y operaciones para trabajar con la estructura.

Actividad 1: TDA punto dominio

- Estructura de datos:

  Un punto P= (x, y) se define por las coordenadas x e y, donde ambas coordenadas son números reales.

- Pseudocódigo:

```
REGISTER punto:
    NUM x
    NUM y
```

Actividad 1: TDA punto operaciones

- **`crea_punto`**`(NUM x, NUM y): punto*`
  - Crea un punto en el plano asignando los valores de las coordenadas **x** e **y.**
  - Devuelve el punto creado.

- **`calcula_distancia`**`(punto *A, punto *B): NUM`
  - Calcula la distancia entre el punto A =(xa,ya) y el punto B=(xb,yb) con la fórmula: d = ((yb-ya)^2 + (xb – xa)^2)^(1/2).
  - Devuelve la distancia obtenida.

- **`distancia_origen`**`(punto *A): NUM`
  - Calcula la distancia entre el punto O =(0,0) y el punto A=(x, y) con la fórmula: d = (x^2 + y^2)^(1/2).
  - Devuelve la distancia obtenida

Actividad 2: implementación TDA

- La implementación se divide en tres archivos:
  1. Una interfaz, que define la estructura de datos y declara prototipos de las funciones que se utilizarán para operar la estructura de datos (.h).
  2. Una implementación de las funciones declaradas en la interfaz (.c).
  3. Un programa cliente que utiliza las funciones declaradas en la interfaz para trabajar en un nivel superior de abstracción (.c).

Actividad 2: implementación TDA

- Implementar en lenguaje *C* el TDA punto en el plano cartesiano

Actividad 2: interfaz .h

- Nombre: TDApunto.h
- Estructura de datos

```
typedef struct{
        float x;
        float y;} punto;
```

- Operaciones

```
// prototipos
punto* crea_punto(float x, float y);

float calcula_distancia(punto *a, punto *b);

float distancia_origen(punto *a);
```

Actividad 2: implementación .c

- Nombre: TDApunto_implementacion.c

```
#include <math.h>
#include <stdlib.h>
#include "TDApunto.h"
```

- **crea_punto**(NUM x, NUM y): punto*
  - Implementación:

```
punto* crea_punto(float x, float y){
      punto *nuevo_punto = malloc(sizeof(punto));
      nuevo_punto->x = x;
      nuevo_punto->y = y;
      return nuevo_punto;
      }
```

Actividad 2: implementación .c

- **calcula_distancia**(punto *A, punto *B): NUM
  - Implementación:

```
float calcula_distancia(punto *a, punto *b){
      float dx = a->x - b->x;
      float dy = a->y - b->y;
      return sqrt(dx * dx + dy * dy);
      }
```

- El archivo de cabecera math.h contiene la función sqrt() #include <math.h>

Actividad 2: implementación .c

- **distancia_origen**(punto *A): NUM
  - Implementación:

```
float distancia_origen(punto *a){
      punto *origen = crea_punto(0.0, 0.0);
      return calcula_distancia(origen, a);
      }
```

Actividad 3a: programa cliente .c

- Implementar en lenguaje *C* un programa que solicite dos puntos (x e y para cada uno) y que muestre por pantalla la distancia entre ellos.

Actividad 3a: programa cliente .c

- Implementar en lenguaje *C* un programa que solicite dos puntos (x e y para cada uno) y que muestre por pantalla la distancia entre ellos.

.

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

int main(void){



    return 0;
}
```

Actividad 3b: operación suma puntos

- Implementar en C la operación suma de puntos:

```
suma_puntos(punto *a, punto *b): punto*
        NUM x ← a→x + b→x
        NUM y ← a→y + b→y
        punto *punto_suma ← crea_punto(x, y)
        return punto_suma
```

1. Declare el prototipo de la función en la interfaz TDApunto.h
2. Implemente la operación en TDApunto_implementación.c
3. Verifique su funcionamiento desde actividad_3.c

Actividad 4: puntos aleatorios

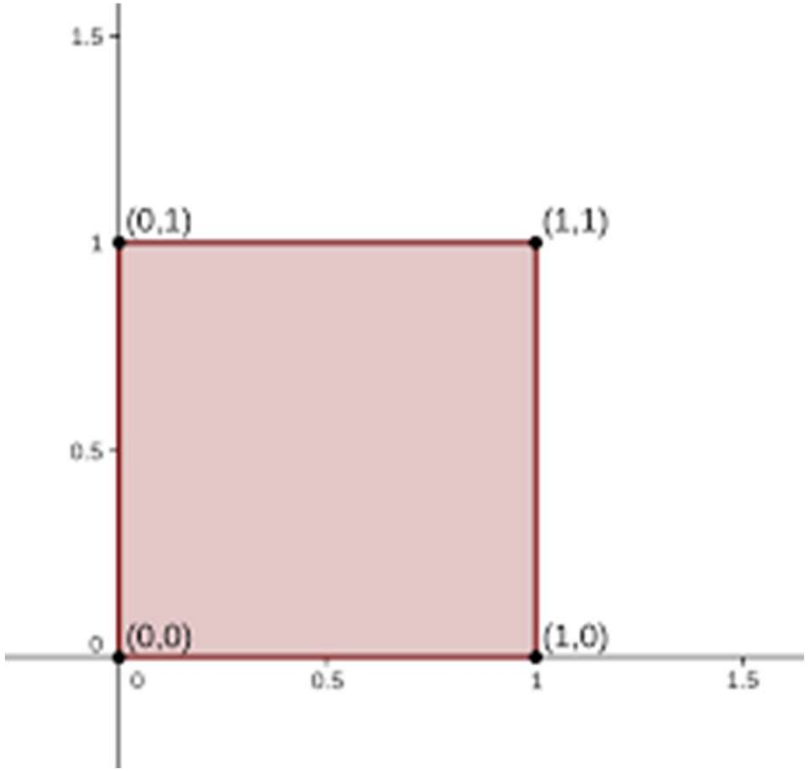
- Diseñe en pseudocódigo un algoritmo que genere aleatoriamente N puntos en un cuadrado unitario y que cuente el número de pares de puntos que se pueden conectar mediante una línea recta de longitud menor que una distancia dada.

- Para su solución considere:
  - Utilice un registro para representar los puntos
  - Asuma la existencia de un procedimiento de nombre rand() que genera un número aleatorio entre 0 y 1

Actividad 4: puntos aleatorios

- Entrada: 10 y 2
- Salida: 45

- Entrada: 10 y 0
- Salida: 0

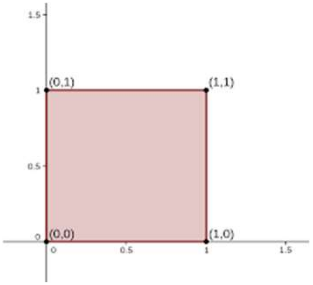


Actividad 4: puntos aleatorios

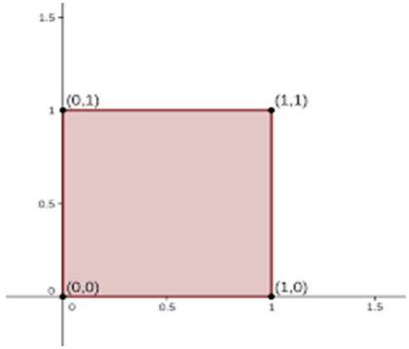
```
puntos_aleatorios(NUM n_puntos, NUM distancia): NUM
   REGISTER punto
      NUM x
      NUM y

   punto A[n_puntos]
   FOR i ← 1 TO n_puntos
      A[i] ← crea_punto(rand(), rand())

   NUM contador ← 0
   FOR i ← 1 TO n_puntos - 1
       FOR j ← i + 1 TO n_puntos
           IF distancia_puntos(A[i], A[j]) < distancia THEN
               contador ← contador + 1
   RETURN contador
```



Actividad 4: puntos aleatorios



- Implementar el algoritmo en C
- Función que genera un número aleatorio entre 0 y 1

```
float flotante_aleatorio(){
   return 1.0 * rand()/RAND_MAX;
   }
```

- Declaración de un arreglo de puntos (punteros a puntos)

```
punto **a = (punto**)malloc(N * (sizeof(punto)));
```

Actividad 5: TDA conjunto

- Compilar y ejecutar lab05-conjunto.c
- Experimentar con las funciones de TDAconjunto, haciendo llamadas desde la función main() en lab05-conjunto.c para realizar lo siguiente:
  1. Crear un conjunto
  2. Mostrar los elementos del conjunto
  3. Agregar el elemento 2
  4. Agregar el elemento 4
  5. Mostrar los elementos del conjunto
  6. Eliminar el elemento 4
  7. Mostrar los elementos del conjunto

Actividad 5: TDA conjunto

- Implementar las siguientes operaciones del TDA conjunto:

- **`union_conjuntos`**`(conjunto *a, conjunto *b): conjunto*`
  - Crea un conjunto que contiene todos los elementos, que pertenecen por lo menos a uno de los conjuntos a o b.
  - Retorna el conjunto creado.

- **`interseccion_conjuntos`**`(conjunto *a, conjunto *b): conjunto*`
  - Crea un conjunto cuyos elementos, necesariamente, pertenecen a ambos conjuntos a y b.
  - Retorna el conjunto creado.

- **`diferencia_conjuntos`**`(conjunto *a, conjunto *b): conjunto*`
  - Crea un conjunto que contiene los elementos de a que no están en b.
  - Retorna el conjunto creado.

- **`diferencia_simetrica`**`(conjunto *a, conjunto *b): conjunto*`
  - Crea un conjunto que contiene a aquellos elementos que pertenecen a cada uno de los conjuntos iniciales, pero no a ambos a la vez.
  - Retorna el conjunto creado.

Actividad 5: TDA conjunto

- Verificar el funcionamiento de las operaciones implementadas con los conjuntos:
  - A={1,2,3,4,5} y B={4,5,6,7,8,9}
  - A∪B={1,2,3,4,5,6,7,8,9}
  - A∩B={4,5}
  - A-B={1,2,3}
  - AΔB={1,2,3,6,7,8,9}

Entrega de actividad de laboratorio

- Entrega obligatoria
- Subir actividades nro. 3b y nro. 5 de esta sesión en buzón de Campus Virtual en un único archivo s5_ apellido_nombre.zip
- Se espera TDApunto.h, TDApunto_implmentacion.c, actividad_3.c, TDAconjunto.h, TDAconjunto_implmentacion.c y lab05-conjunto.c comprimidos en archivo .zip
- Plazo: hoy dentro del horario de laboratorio de cada coordinación

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Actividad de cierre

¿Completó las actividades síncronas?

¿Qué dificultades encontró?