UNIVERSIDAD INCA GARCILASO DE LA VEGA
FACULTAD DE INGENIERIA DE SISTEMAS, COMPUTO Y TELECOMUNICACIONES

ASIGNATURA	Estructura de Información
TEMA	Tipos Abstractos de Datos
PROFESOR	Carlos A. Ruiz De La Cruz Melo
ALUMNO	
CODIGO	
FECHA	
CICLO	
TURNO	
SEMESTRE	2010-II

1. OBJETIVOS

Que el estudiante:

- Aprenda a especificar los tipos abstractos de datos (TAD)
- Aprenda a implementar los TAD

2. INTRODUCCION TEORICA

ESTRUCTURA DE DATOS

Es un conjunto de variables de un determinado tipo agrupadas y organizadas de alguna manera para representar un comportamiento. Lo que se pretende con las estructuras de datos es facilitar un esquema lógico para manipular los datos en función del problema que haya que tratar y el algoritmo para resolverlo. En algunos casos la dificultad para resolver un problema radica en escoger la estructura de datos adecuada.

Distinguimos las estructuras de datos siguientes:

- Filas secuenciales
- Arreglos
- Listas enlazadas
- Pilas y colas
- Árboles

TIPOS DE DATOS

Es la especificación de un conjunto de valores y de un conjunto valido de operaciones que se asignan sobre una variable.

TIPOS DE DATOS SIMPLES

Son los tipos de datos comunes o básicos como enteros, reales, cadenas alfanuméricas, etc

TIPOS COMPUESTOS O TIPOS ABSTRACTOS DE DATOS(TAD)

Un tipo compuesto permite describir una estructura de datos como un conjunto de valores junto con las operaciones que sobre el se pueden aplicar, las cuales cumplirán diversas propiedades que determinarán su comportamiento.

Características de un TAD

- Con los TAD se quiere plasmar lo esencial de la realidad sin comprometerse con detalles de implementación(su desarrollo es independiente del lenguaje de programación). De hecho, es posible considerar diferentes implementaciones
- Un TAD es una estructura algebraica, o sea, un conjunto de objetos con ciertas operaciones definidas sobre ellos. Piense, por ejemplo, en la calculadora: los elementos que maneja son cantidades numéricas y las operaciones que tiene definidas sobre éstas son las operaciones aritméticas. Otro ejemplo posible es el TAD matriz; los elementos que maneja son las matrices y las operaciones son aquellas que nos permiten crearlas, sumarlas, invertirlas, etc.
- A la hora de utilizar el TDA, la representación debe permanecer oculta. Solo podremos utilizar las operaciones del tipo para trabajar con sus elementos.
- ENCAPSULAMIENTO: Se desconoce la implementación de la declaración y de las operaciones del TAD.

TIPOS BASICOS DE OPERACIONES EN UN TAD

Es posible observar que las operaciones de un TAD son de diferentes clases: algunas de ellas nos deben permitir crear entidades nuevas, otras determinar su estado o valor en un instante determinado, unas construir otras entidades a partir de algunas ya existentes, etc.

Aquí los tipos básicos

- **Constructores**: Crean una nueva instancia del tipo.
- **Transformación**: Cambian el valor de uno o más elementos de una instancia del tipo.
- **Observación**: Nos permiten observar el valor de uno o varios elementos de una instancia sin modificarlos.
- **Iteradores**: Nos permiten procesar todos los componentes en un TDA de forma secuencial.

ESPECIFICACIÓN E IMPLEMENTACIÓN DE TADs

Como ocurre con toda abstracción, tanto de operaciones como de datos, debemos distinguir dos niveles en su **DISEÑO**:

1.- Especificación o definición ¿*qué es*? o ¿*qué hace*? La abstracción. Esta definición puede ser formal o informal. *Ejemplo:* Especificación de una abstracción de operaciones

mediante su pre/postcondición expresadas mediante el lenguaje lógico.

2.- Implementación o ¿*Cómo es*? o ¿*Cómo lo hace*? en un determinado entorno informático. *Ejemplo:* Implementación de una abstracción de operaciones mediante las “function” y “procedure” de los lenguajes de programación.

ESPECIFICACIÓN DE UN TIPO DE DATO

Una especificación algebraica consta fundamentalmente de tres componentes**:**

- **Nombre del tipo :** especifica el nombre del TAD
- **Usar** : declara si se utilizan otros TADs previamente definidos
- **Variable:** atributo que contiene el estado(valor) del TAD
- **Lista de operaciones** : especifica la sintaxis o perfil de las operaciones. Se dará como cabecera de procedimiento o función
- **Ecuaciones** : es la semántica de las operaciones sobre el tipo: precondición y postcondición

Por ejemplo, podemos describir los booleanos de la siguiente forma

Especificación Booleanos
variable
b: bool
operaciones
cierto: → bool
falso: → bool
bool ∧ bool → bool
bool ∨ bool → bool
no bool → bool
significado
(cierto ∧ b) = b
(falso ∧ b) = falso
(cierto ∨ b) = cierto
(falso ∨ b) = b
no (cierto) = falso
no (falso) = cierto
Fin_Boleanos

Por ejemplo, podemos describir los números naturales de la siguiente forma

Especificación NATURALES
usar Boleanos
variable
num: natural
operaciones
cero : → natural
suc : natural → natural
+,-,* : natural natural →natural
Exp : natural natural → natural
==, ≠ : natural natural → bool

significado
Cero + m = m
Suc(n) + m = suc(n+m)
Cero*m = cero

Fin_NATURALES

VENTAJAS DE USAR TADs:

- Independencia de la implementación del tipo.
- Programas más portables, legibles, reutilizables.
- Mayor seguridad de la información, menos efectos colaterales.
- Mayor facilidad para comprobar la corrección.

3. REQUERIMIENTOS O MATERIAL Y EQUIPO

- Software Dev C++
- 1 Diskete

4. PROCEDIMIENTO

El procedimiento consiste en dos pasos

- Especificación del TAD
- Implementación del TAD

Ejercicio 1.

Se desea un programa que permita tomar los datos de un alumno y obtener el promedio de las notas, el cual se obtiene de promediar el examen parcial, examen final y promedio de practicas. El programa debe mostrar los resultados

Especificación

```
Especificación ALUMNO
variable
        entero   : codigo
        cadena : cadena
        real     : exa_parcial
        real     : exa_final
        real     : prom_practicas
operaciones
        IngresarAlumno              : no retorna valor
        MostrarAlumno               : no retornar valor
        ObtenerPromedio(promedio) : retorna valor lógico
significado
        En ObtenerPromedio, el promedio debe ser mayor o igual a una kte(10.5)
Fin_ALUMNO


función ObtenerPromedio(promedio): lógico
     real: promedio
    promedio ←(exa_parcial+exa_final+prom_practicas)/3
    si (promedio>=10.5) entonces
        obtener← verdadero
      sino
         obtener← falso
    fin_si
    retornar obtener
fin_ObtenerPromedio
```

Implementación

```
#include “iostream.h”
#include “conio.h”
class ALUMNO{
    int codigo;
    char nombre[50];
    float exa_parcial;
    float exa_final;
    float prom_practicas;
public:

  ALUMNO(){
    codigo=0;
    strcpy(nombre,””);
    exa_parcial=0;
```

```
  exa_final=0;
  prom_practicas=0;
        }
void IngresarAlumno(){
   cout<<"\n leer codigo : ";cin>>codigo;
   cout<<"\n leer nombre : ";cin>>nombre;
   cout<<"\n leer parcial: ";cin>>exa_parcial;
   cout<<"\n leer final  : ";cin>>exa_final;
   cout<<"\n leer pp     : ";cin>>prom_practicas;
  }
void MostrarAlumno(){
   float promedio;
   cout<<"\n "<<codigo<<"      "<<nombre;
   cout<<"\n parcial :"<<exa_parcial<<"\n final: "<<exa_final<<"\n PP: "<<prom_practicas;
   if(ObtenerPromedio(promedio)){
       cout<<"\n El alumno aprueba con nota ";
      cout.precision(4);
       cout<<promedio;
   }
   else cout<<"\n El alumno desaprueba con nota "<<promedio;
   }
bool ObtenerPromedio(float &promedio){
   promedio=(exa_parcial+exa_final+prom_practicas)/3;
   if(promedio>=10.5) return true;
    else return false;
  }
   };

main(){
   ALUMNO a;
   a.IngresarAlumno();
   a.MostrarAlumno();
   getch();
       }
```

Ejercicio 2.

Se desea un programa que haga lo mismo que el anterior solo que la especificación así como la implementación cambiaran.

Especificación

Especificación NOTA
variable
real : num
operaciones
RegistrarNota : no retorna valor
RetornarNota : real
significado
// explicación de las operaciones
Fin_NOTA

Especificación ALUMNO
Usar : NOTA
variable
entero : código
cadena : nombre
NOTA : lista

operaciones

IngresarAlumno : no retorna valor
MostrarAlumno : no retornar valor
ObtenerPromedio(promedio) : promedio >= kte → booleano

Ecuaciones

El promedio debe ser mayor o igual a una kte(10.5)

Fin_ALUMNO

```
función ObtenerPromedio(promedio): lógico
        real : promedio
        promedio ←0
        i←0
        mientras (i<3)  hacer
              promedio ←promedio + n[i].retornarnota()
              i ← i + 1
           fin_mientras
        promedio ←promedio/3
        si(promedio >=10.5) entonces
                 obtener ← verdadero
          sino  obtener ← falso
        fin_si
        retornar obtener
  finObtenerPromedio
```

Implementación

```
#include "iostream.h"
#include "conio.h"

class NOTA{
  float num;
  public:
  NOTA(){ num=0; }
  void RegistrarNota(char mensaje[]){
    cout<<"\n "<<mensaje<<" : ";cin>>num;
  }
  float RetornarNota (){ return num; }
};


class ALUMNO{
    int codigo;
    char nombre[50];
    NOTA n[3];
  public:
    ALUMNO(){
      codigo=0;
      strcpy(nombre,"");
    }
    void IngresarAlumno(){
        char tira[5],mensaje[20];

        cout<<"\n leer codigo : ";cin>>codigo;
        cout<<"\n leer nombre : ";cin>>nombre;
        for(int i=0; i<3; i++){
           strcpy(mensaje,"nota ");
           itoa(i,tira,10);
```

```
            strcat(mensaje,tira);
            n[i].RegistrarNota(mensaje);
        }
     }
    void MostrarAlumno(){
        float promedio;
        cout<<"\n "<<codigo<<"      "<<nombre;
        cout.precision(4);
        if(ObtenerPromedio(promedio)){
           cout<<"\n El alumno aprueba con nota ";
           cout<<promedio;
           }
        else cout<<"\n El alumno desaprueba con nota "<<promedio;
   }
   bool ObtenerPromedio(float &promedio){
       promedio=0;
       for(int i=0; i<3; i++)
           promedio=promedio + n[i]. RetornarNota ();
       promedio=promedio/3;
       if(promedio>=10.5) return true;
       else return false;
   }
};

main(){
   ALUMNO a;
   a.IngresarAlumno();
   a.MostrarAlumno();
   getch();
 }
```

Ejercicio 3.

En la Clínica Odontológica de la Universidad Inca Gracilazo de la Vega trabajan docentes y alumnos. Cada docente es el tutor de un grupo de alumnos y los alumnos solo pueden estar supervisados por un tutor, los cuales al ser parte de la clínica tienen que estar registrados con el tutor que deseen, por supuesto que el registro lo hace el tutor.

Ejercicio 4.

En una empresa de telecomunicaciones se forman equipos de periodistas para realizar reportajes. Cada equipo de periodistas esta formado por dos periodistas, un camarógrafo y un relator de noticias. La empresa cuenta con número limitado de periodistas, así que el programa debe verificar que los periodistas que forman los equipos sean trabajadores de la empresa. Cada reportaje tiene fecha de inicio, titulo, y tiempo de duración.
El programa debe generar un reporte en el cual dado el nombre o código de un periodista muestre los reportajes en los que esta trabajando.

Ejercicio 5.
Una tienda de libros registra la venta efectuada a sus clientes en un documento(boleta) el cual tiene como información los datos del cliente, datos de la venta y datos del vendedor para efectos de comisiones. Elabore un reporte en el cual dado el nombre o código de un vendedor permita ver las ventas efectuadas por ese vendedor.

5. ANALISIS DE RESULTADOS

- Los TDAs permiten la creación de instancias con propiedades bien definidas y comportamiento bien definido.
- En orientación a objetos, nos referimos a los TDAs como *clases*. Por lo tanto, una clase define las propiedades de *objetos* instancia en un ambiente orientado a objetos.
- Los TDAs definen la funcionalidad al poner especial énfasis en los datos involucrados, su estructura y operaciones.
- Consecuentemente, la programación orientada a objetos es "programación con TDAs" : al combinar la funcionalidad de distintos TDAs para resolver un problema.
- Por lo tanto, instancias (objetos) de TDAs (clases) son creados dinámicamente, usados y destruidos.

6. BIBLIOGRAFIA

- Estructura de Datos en Pascal, AARON M. TENENBAUM, MOSHE J. AUGENSTEIN
- Fundamentos de Programación, Algoritmos y Estructura de Datos. LUIS JOYANES AGUILAR
- Estructura de Datos y Diseño de Programas. ROBERT L KRUSE
- Estructura de Datos, Un Enfoque Algorítmico. Manuel Gallardo O., Teodomiro Pérez C.