**Instituto Superior de Formación Técnica Nº 151 logo151-trans.gif  
Carrera: Analista de Sistemas  
1 Año. Algoritmos y Estructuras de Datos I**

|  |  |
| --- | --- |
| **Trabajo Práctico Nº 5** | **Unidad 5** |
| **Modalidad:** Semi-Presencial | **Estratégica Didáctica:** Trabajo Grupal. |
| **Metodología de Desarrollo:** acordar | **Metodología de Corrección:** acordar docente |
| **Carácter de Trabajo:** Obligatorio – con Nota | **Fecha Entrega:** A confirmar por el Docente. |

**Marco Teórico:**

1. Describir el Concepto de Array
2. Dar un Ejemplo de:
   1. Creación de Arrays (preferentemente con typedef)
   2. Asignación de Arrays
   3. Recorrido de Arrays.
3. Describir el Concepto de Listas (sin estructura)
   1. Dar un ejemplo de Inserción de un elemento
   2. Dar un ejemplo de Borrado de un elemento
4. Describir el Tipo de Dato “String”
   1. Dar ejemplos de 4 Funciones “Incluidas” en el Tipo “String”
   2. Que sentencia de Entrada usaría para leer un String con espacio x teclado
5. Estructuras de Datos
   1. ¿Qué entiende por Estructuras de Datos (struct)?
   2. Dar 4 ejemplos de Estructuras.
   3. Dar un Ejemplo de cómo se definiría una Estructura para que guarde sus datos personales más relevantes.
   4. ¿Cuál es el sentido de tener Estructuras dentro de Estructuras?
   5. Dar un ejemplo para el caso del punto 5.4 en código
   6. ¿Qué ventaja tiene armar una lista con estructuras?
   7. Dar un ejemplo en código.
   8. ¿Qué operaciones podemos tener en la lista de Estructura?
   9. Dar un ejemplo en código de lo respondido en el punto Anterior.
6. Describir la sentencia de repetición Do While
   1. Qué diferencia tiene con la sentencia While
   2. Dar 2 ejemplos de Uso de la Sentencia.

**Marco Practico:**

**Desarrollar un Programa** que mantenga una lista de los estudiantes de Algoritmos y Estructuras de Datos I.

**Se deberán tener en cuenta las siguientes particularidades:**

1. De cada estudiante: nombre, apellidos, edad, DNI y nota
2. Se desconoce el número total de estudiantes (máximo 50)
3. La información de la lista se mantiene en un archivo AyEDI.txt
4. Se carga el Archivo en Memoria llenando la Estructura al empezar y se guarda la estructura al finalizar el Programa en el mismo Archivo sobrescribiéndolo.  
     
   **El programa debe ofrecer estas opciones:**
5. Añadir un nuevo alumno
6. Eliminar un alumno existente
7. Calificar a los estudiantes
8. Listado de notas, identificando la mayor y la media

**Para un mejor desarrollo tener en cuenta:**

1. Deberán realizar una Estructura para la lista que contenga el array, cantidad de alumnos y estructura del alumno.
2. Deberán priorizar la Modularización (mediante funciones que permitan: Cargar, Leer, guardar e insertar alumnos).
3. Deberán tener una Función para leer el Archivo cuando arranca el programa y otra para guardar la estructura de los Alumnos, se lee todo el archivo cuando arranca, se modifica en memoria (las estructuras) y se guardan todos los Alumnos sobrescribiendo el archivo original.
4. Para el menú un Do..While con las opciones Solicitadas, que deberán llamar a las Funciones Desarrolladas, recibiendo como parámetro la Estructura, cantidad de elementos y demás.
5. Trate de mantener el “Nivel de Abstracción” desde lo General a lo Particular, es decir, no arranque a Codificar, Piense en “Macro” (que debe hacer el Programa) … Estructure lo que hace en Funciones, modularizando el programa y Minimizando el Main().
6. La función Principal (Main) deberá contener el menú y las llamadas a las Funciones y todo el código de Definición de Variables.
7. El problema se resuelve con los prácticos anteriores y en caso de necesitar revea el material visto en clase, debería ser suficiente para desarrollarlo.

Lic. Oemig José Luis.

1. Describir el Concepto de Array

Un Array es una colección homogénea de datos, es decir, todos de un mismo tipo, que están organizados y son accedidos mediante un índice: 0, 1, 2, …

1. Dar un Ejemplo de:
2. Creación de Arrays (preferentemente con typedef)

*const int Dimension = 20;*

*typedef double tCompras[Dimension];*

1. Asignación de Arrays:

*tCompras arreglo1, arreglo2;*

*// Supongamos que el arreglo1 tiene determinado contenido...*

*for (int i = 0 ; i < Dimension ; ++i)*

*{*

*arreglo2[i] = arreglo1[i];*

*}*

1. Recorrido de Arrays

*tCompras arreglo1, arreglo2;*

*// Supongamos que el arreglo1 tiene determinado contenido.*

*// Vamos a recorrerlo y mostrar su contenido por pantalla.*

*for (int i = 0 ; i < Dimension ; ++i)*

*{*

*cout << arreglo1[i] << endl;*

*}*

3. Describir el Concepto de Listas (sin estructura)

Una **lista** es una colección de elementos homogéneos de manera que se establece entre ellos un orden *lineal*. Es decir, cada elemento, menos el primero, tiene un predecesor, y cada elemento, menos el último, tiene un sucesor.

3.1. Dar un ejemplo de Inserción de un elemento

Resuelto en el archivo *Punto\_3.cpp*

3.2. Dar un ejemplo de Borrado de un elemento

Resuelto en el archivo *Punto\_3.cpp*

4. Describir el Tipo de Dato “String”

Un *string* o cadena de caracteres es una secuencia de caracteres de longitud variable.

4.1 Dar ejemplos de 4 Funciones “Incluidas” en el Tipo “String”

Hay diversos métodos (funciones miembro) asociados al tipo *string*:

1. length() : devuelve la longitud de la cadena.

Ejemplo:

#include<string>

#include<iostream>

using namespace std;

int main()

{

string cadena1 = "Hola";

cout << "cadena:" << cadena1;

cout << "\n";

cout << cadena1.size();

cout << "\n";

cout << cadena1.length();

return 0;

}

1. empty() : función booleana: devuelve *true* si la cadena es vacía y *false* en caso contrario

Ejemplo:

#include<string>

#include<iostream>

using namespace std;

int main()

{

string cadena;

cout << "Ingrese una cadena:";

getline(cin, cadena);

if (cadena.empty())

{

cout << "No se ingresaron caracteres por teclado.";

}

else

{

cout << "La cadena tiene una longitud de:" << cadena.length();

}

return 0;

}

Otras funciones:

#include <iostream>

#include <string>

using namespace std;

int main() {

string cad1("Hola"); // inicialización

string cad2 = "amigo"; // inicialización

string cad3;

cad3 = cad1; // copia

cout << "cad3 = " << cad3 << endl;

cad3 = cad1 + " "; // concatenación

cad3 += cad2; // concatenación

cout << "cad3 = " << cad3 << endl;

cad1.swap(cad2); // intercambio

cout << "cad1 = " << cad1 << endl;

cout << "cad2 = " << cad2 << endl;

return 0;

}

4.2 ¿Qué sentencia de Entrada usaría para leer un String con espacio x teclado?

getline(cin, cadena); // lee incluyendo espacios en blanco.

5. Estructuras de Datos

5.1 ¿Qué se entiende por Estructuras de Datos (struct)?

Un *struct* es un tipo de dados estructurado con un número fijo de componentes a los que se tiene acceso por medio de un nombre. Los componentes pueden ser heterogéneos (de tipos distintos). El acceso a dichos componentes se efectúa mediante el operador ‘.’ :

*nombre\_de\_la\_estructura.nombre\_del\_componente*

5.2 Dar 4 ejemplos de Estructuras.

Ejemplo 1:

***typedef struct*** *{*

***string*** *nombres;*

***string*** *apellidos;*

***int*** *edad;*

***int*** *dni;*

*} tUsuario;*

Ejemplo 2:

***typedef struct*** *{*

***string*** *autor;*

***string*** *tema;*

***int*** *duracion;*

***int*** *anio;*

*} tAlbum;*

Ejemplo 3:

***const int*** *MAX = 100;*

***typedef struct*** *{*

***double*** *arreglo[MAX];*

***int*** *tamanio;*

*} tLista;*

Ejemplo 4:

***typedef struct*** *{*

***string*** *nombreCliente;*

***string*** *razonSocial;*

***int*** *numFactura;*

***double*** *monto;*

*} tVentas;*

5.3 Dar un Ejemplo de cómo se definiría una Estructura para que guarde sus datos personales más relevantes.

***typedef struct*** *{*

***string*** *nombres;*

***string*** *apellidos;*

***int*** *edad;*

***int*** *dni;*

***string*** *domicilio;*

***int*** *telefono;*

*} tUsuario;*

5.4 ¿Cuál es el sentido de tener Estructuras dentro de Estructuras?

El principal motivo para tener Estructuras dentro de Estructuras es la de poder reutilizarlas mediante la Composición de las mismas.

5.5 Dar un ejemplo para el caso del punto 5.4 en código

***typedef struct*** *{*

***int*** *dia;*

***int*** *mes;*

***int*** *anio;*

*} tFecha;*

***typedef struct*** *{*

*tFecha nacimiento;*

***string*** *nombres;*

***string*** *apellidos;*

***int*** *dni;*

***string*** *domicilio;*

***int*** *telefono;*

*} tUsuario;*

5.6 ¿Qué ventaja tiene armar una lista con estructuras?

Una de las ventajas es la de poder destinar un campo para la lista en sí y otro campo (entero o enumerado) destinado a controlar el tamaño.

5.7 Dar un ejemplo en código.

***const int*** *N = 10;*

***typedef double*** *tArray[N];*

***typedef struct*** *{*

*tArray elementos;*

***int*** *contador;*

*} tLista;*

5.8 ¿Qué operaciones podemos tener en la lista de Estructura?

En una lista se pueden tener asociadas operaciones de inserción, borrado, copiado, concatenación, búsqueda, ordenación, etc.

5.9 Dar un ejemplo en código de lo respondido en el punto Anterior.

Ver archivo ***Punto\_5\_9.cpp***

6. Describir la sentencia de repetición Do While

*do {*

*cuerpo*

*} while (condición);*

Esta sentencia ejecuta el cuerpo mientras la condición es verdadera.

* 1. ¿Qué diferencia tiene con la sentencia While?

La principal diferencia radica en que el While chequea la condición antes de ejecutar el cuerpo, mientras que el do..While chequea la condición al final.

* 1. Dar 2 ejemplos de Uso de la Sentencia.

Ejemplo 1:

*int op;*

*cout << "Proveedores:" << endl;*

*cout << endl;*

*cout << "1. Los Amigos" << endl;*

*cout << "2. Maritn y Carolina" << endl;*

*cout << "3. Juan Jose" << endl;*

*cout << "4. Los Rodriguez" << endl;*

*cout << "5. La Huerta" << endl;*

*cout << endl;*

*cout << "Elija un proveedor: ";*

*do*

*{*

*cin >> op;*

*}*

*while ((op < 1) && (op > 5) );*

Ejemplo 2:

*int op;*

*do*

*{*

*cout << endl;*

*cout << "Desea pedir? ( 1 = SI / 0 = NO ): ";*

*cin >> op;*

*}*

*while((op != 0) && (op != 1) );*