Universidad de Panamá

Vicerrectoría de Investigación y Postgrado

Facultad de Informática, Electrónica y Comunicación

Maestría en Ingeniería de Sistemas E-Learning

Repositorio y Metadatos de Objetos de Aprendizaje

ACTIVIDAD 3

"Transformación de una actividad de aprendizaje desde una modalidad presencial a una modalidad virtual"

A consideración de:

Profesora Cindy Esquivel

Presentado por:

SAULO AIZPRUA A. **6-87-480**

Panamá, Mayo 2019

Índice

1.OA- Arreglos Unidimensionales

ARREGLOS UNIDIMENSIONALES

Arreglos Unidimensionales

Descripción

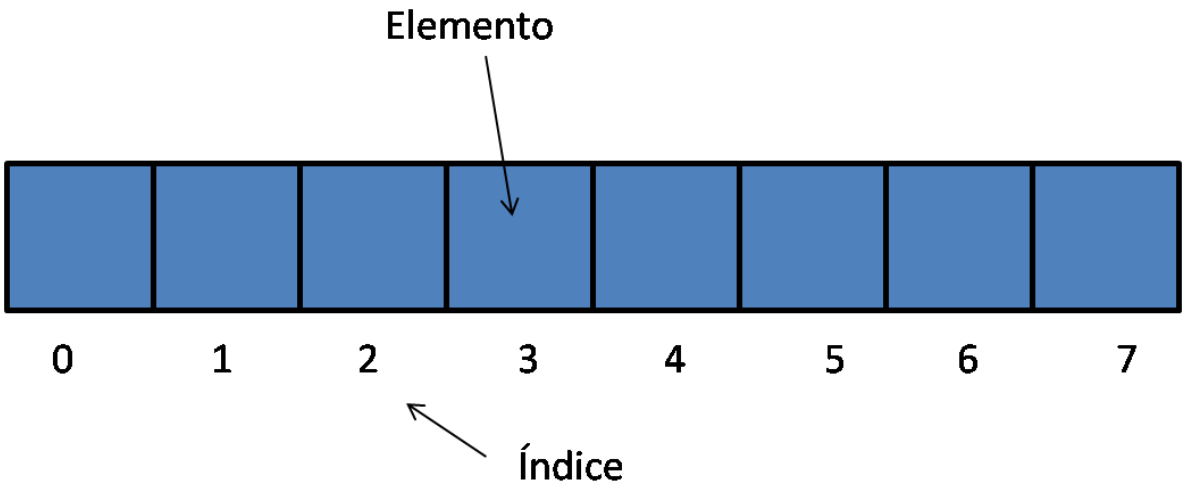
DESCRIPCIÓN

Arrays, arreglos o vectores en C++

Los arrays, arreglos o vectores forman parte de la amplia variedad de estructuras de datos que nos ofrece C++, siendo además una de las principales y más útiles estructuras que podremos tener como herramienta de programación. Los arrays, arreglos o vectores (como los quieras llamar), son utilizados para almacenar múltiples valores en una única variable. En un aspecto más profundo, los arrays, permiten almacenar muchos valores en posiciones de memoria continuas, lo cual permite acceder a un valor u otro de manera rápida y sencilla. Estos valores pueden ser números, letras o cualquier tipo de variable que deseemos incluso tipos de datos propios.

En múltiples ocasiones es necesario almacenar gran cantidad de información en una variable y a menudo sucede que no conocemos con exactitud la cantidad de datos que debemos almacenar, pero sabemos que sí sería más de uno, como por ejemplo almacenar las identificaciones de las personas ingresadas al sistema. Los arrays, arreglos o vectores son una estructura que nos permite solucionar este tipo de problemas.

Un array o arreglo en C++ es un conjunto de datos que se almacenan en memoria de manera seguida con el mismo nombre. Es una colección de variables del mismo tipo, cada una de ellas se llama elemento y posee una posición dentro del arreglo llamado índice.



A quien va Dirigido

Este Objeto de aprendizaje esta dirigido para todos los estudiantes de las Facultad de Informática, Electrónica y Comunicación de la Universidad de Panamá , del primer años del curso de PROGRAMACIÓN I, de las siguientes carreras: Licenciatura en Ingeniería en Informática, Licenciatura en Ingeniería en Electrónica y Comunicación, Licenciatura en Comercio Electrónico, Licenciatura en Informática Aplicada a la Educación, Licenciatura en Informática Aplicada a la Gestión Empresarial.

Objetivos

- Al finalizar este objeto de aprendizaje el estudiante será capaz de identificar que es un arreglos y su ubicación dentro de los tipos de estructura de datos.
- El estudiantes clasificará los diferentes tipos de arreglos, su uso y comprender de qué manera acceden a la memoria.

- Manejará los compiladores de C++ para realizar sus prácticas, laboratorios e implementar operaciones con arreglos.
- Es estudiante desarrollará en lenguaje de programación C++ las operaciones sobre un arreglo unidimensional.

Contenidos

A- ARREGLOS UNIDIMENSIONALES

A.1 DEFINICIÓN

Un arreglo unidimensional tiene una sola dimensión. A los arreglos de una dimensión también se les llama listas.

Un arreglo unidimensional es un tipo de datos estructurado que está formado de una colección finita y ordenada de datos del mismo tipo. Es la estructura natural para modelar listas de elementos iguales.

El tipo de acceso a los arreglos unidimensionales es el acceso directo, es decir, podemos acceder a cualquier elemento del arreglo sin tener que consultar a elementos anteriores o posteriores, esto mediante el uso de un índice para cada elemento del arreglo que nos da su posición relativa como se muestra en la Figura 1.

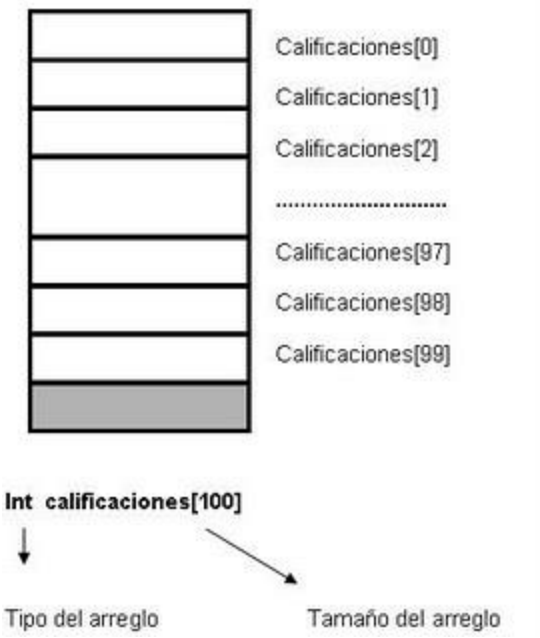


Fig1.

Para implementar arreglos unidimensionales se debe reservar espacio en memoria, y se debe proporcionar la dirección base del arreglo, la índice superior y el inferior.

Operaciones

Para manejar un arreglo, las operaciones a efectuarse son:

- Declaración del arreglo,
- Creación del arreglo,
- Inicialización de los elementos del arreglo.
- Acceso a los elementos del arreglo.

Mostraremos un video de apoyo al contenido de los arreglos unidimensionales

A.2 DECLARACIÓN

¿Cómo declarar un Array o Vector en C++?

Para declarar un vector en C++, se deben seguir las mismas normas básicas que se siguen para declarar una variable cualquiera, con un pequeño cambio en la sintaxis. Para declarar un vector, arreglo o como lo quieras llamar, necesitaremos saber el tipo de los datos que irán al interior de este, es decir, serán número enteros, o número decimales o cadenas de texto, etc. necesitamos también, como siempre,

un nombre para el vector y un tamaño máximo. La sintaxis para declarar un vector en C++ es la siguiente figura Fig2.:

```
tipo_de_dato nombre_del_vector[tamanio];
```

Fig3.

Tenemos entonces, tal como mencioné antes, que para declarar un vector en C++, debemos definirle un tipo de los datos, sea entero, float, string, etc., debemos darle un nombre y al interior de los corchetes "[]" debemos poner el tamaño máximo que tendrá el vector, es decir la cantidad máxima de datos que podrá contener (recuerda que en C++ esto es necesario hacerlo). Veamos un ejemplo en el cual pondré la declaración de varios vectores de diferentes tipos y tamaños en C++.

Declaración de un Array o Vector en C++

```
int my_vector1[10];
float my_vector2[25];
string my_vector3[500];
bool my_vector4[1000];
char my_vector5[2];
```

Fig4.

Veamos rápidamente que representa cada línea del código anterior.

Línea 1

Esta línea contiene la declaración de un vector llamado my_vector1, el cual contendrá un máximo de 10 elementos de tipo entero.

Línea 2

Esta línea contiene la declaración de un vector llamado my_vector2, el cual contendrá un máximo de 25 elementos de tipo float.

Línea 3

Esta línea contiene la declaración de un vector llamado my_vector3, el cual contendrá un máximo de 500 elementos de tipo string.

Línea 4

Esta línea contiene la declaración de un vector llamado my_vector4, el cual contendrá un máximo de 1000 elementos de tipo booleano.

Línea 5

Esta línea contiene la declaración de un vector llamado my_vector5, el cual contendrá un máximo de 2 elementos de tipo char.

A.3 INICIALIZACIÓN

¿Cómo inicializar un Array o Vector en C++?

En cuanto tenemos declarado un vector, es posible asignarle valores, evidentemente estos valores deben coincidir con el tipo de dato que le asignamos a dicho vector, no tendría sentido ingresar como valores de un vector cadenas de caracteres si el tipo de dato de dicho vector es numérico.

Voy a mostrar a continuación formas distintas de inicializar un vector, todas son válidas, ya es cuestión de nuestras necesidades y conocimientos determinar cuál es útil y en qué momento. Veamos entonces:

Forma 1 de declarar un Array o Vector en C++

```
string vector[5] = {"5", "hola", "2.7", "8,9", "adios"};
```

Fig5.

Aquí hemos declarado un vector de tipo string tamaño 5 y lo hemos inicializado con diferentes valores, es necesario notar que cada valor va entre comillas dobles "" puesto que son strings. El valor inicial corresponde a la casilla o índice 0 y tiene el valor de "5", el índice 1 el valor es "hola" y el índice 4 el valor es "adiós", es importante notar que el primer índice de n array o vector no es el UNO sino que es el CERO.

Forma 2 de declarar un Array o Vector en C++

```
int vector2[] = {1,2,3,4,10,9,80,70,19};
```

Fig6.

Aquí hemos declarado un vector de tipo int y no especificamos su tamaño, si el tamaño no se especifica entre los corchetes, el vector tendrá como tamaño el número de elementos incluidos en la llave, para este caso es 9.

Particularidades de los Arrays, arreglos o Vectores en C++

Con C++, existen algunas particularidades, en cuanto a la declaración de vectores, que me parece importante destacara para que en momento de quizá caer en ellas comprender como podrían cambiar las cosas o básicamente en que consiste el error, veamos:

Particularidad 1 al momento de declarar o inicializar un Vector o Array en C++

```
int vector2[3];
vector2[3] = {1,5,10};
```

Fig7.

Dadas las características de C++, es fácil pensar que és factible crear o declarar un vector de un tamaño cualquiera y posteriormente inicializarlos de forma habitual como se muestra en este código, sin embargo hacer esto es un error, si declaramos un vector y no lo inicializamos inmediatamente, no es posible inicializarlo de la forma que hemos visto, es decir entre llaves cada valor, como en la línea 2 del código anterior. La única forma de inicializar el vector, o mejor dicho, darle valores a cada una de sus casillas, es hacerlo uno por uno, es decir darle un valor a la casilla cero a la uno y a la 2 (para un vector de tamaño 3). Por defecto, al declarar un vector sin ser inicializado, cada una de las casillas de este vector toma como valor el valor por defecto del tipo de variable, para el caso de los enteros (int) es -858993460. Así entonces para asignar valores a cada casilla lo hacemos así:

```
int vector2[3];
vector2[0] = 1;
vector2[1] = 3;
vector2[2] = 10;
```

Fig8.

Es importante notar en este código, que el número que va entre corchetes ya no indica tamaño (pues vector2 ya está declarado) sino que indica el índice o el numero de la casilla con la cual estaremos operando (recordemos que el primer índice es cero y no uno), en el código anterior, habíamos declarado un vector de tamaño 3, por lo cual debíamos asignar valores a los índices 0, 1 y 2.

Particularidad 2 al momento de declarar o inicializar un Vector o Array en C++

```
float vector3[5] = {10.5};
```

Fig9.

En C++ a la hora de inicializar un array, arreglo o Vector, estamos acostumbrados a que si inicializamos inmediatamente después de declarar el vector, debemos poner la misma cantidad de elementos al interior de las llaves de manera que corresponda con el tamaño del vector, pues bien, estos es lo más recomendable, sin embargo si ponemos una cantidad de elementos menor a la del tamaño real del vector, estamos queriendo decir que estos elementos toman los valores puestos entre las llaves y los demás serian cero, para el caso del código anterior el primer elemento (el del índice cero) va a tener un valor de 10.5 y los otros 4 elementos van a valer cero.

Ya tenemos claro cómo declarar un array o vector en C++, algunas características un tanto particulares de estos, sin embargo aun no sabemos cómo obtener los datos de un array, es decir una vez el array o vector este lleno con los elementos que queremos, como podemos obtener esa información y más aun, como obtener un valor específico dentro del array.

Obtener el valor de una casilla específica en un array en C++

Es muy común el caso en el que tenemos un vector con una enorme cantidad de elementos, sin embargo de todos estos, solo nos interesa uno en especial y corremos con la suerte de saber cuál es su índice, sabiendo el índice de un elemento en un array es bastante sencillo obtener el valor de este:

```
float vector4[5] = {10.5, 5.1, 8.9, 10, 95.2}; //Array con 5 elementos
float numero5 = vector4[4]; //Para acceder al elemento 5, se usa el índice 4
float primerNumero = vector4[0]; //Para el primer elemento se usa el índice 0
```

Fig10.

Como podemos ver, para acceder a un valor específico conociendo el índice del elemento, solo basta con escribir dicho índice entre los corchetes "[ ]", recuerda que el índice comienza desde cero, así por lo tanto en un vector de 5 elementos (como

el del ejemplo), el último elemento esta en el índice 4 y el primer elemento del array en el índice 0.

Ya tenemos entonces una forma de acceder de forma individual a un elemento de un array o vector, vamos a ver ahora como recuperar todos los elementos de un vector de forma simple

A.4 RECORRER EL ARREGLO

Recorrer un Array o Vector en C++

Para obtener todos los datos que se encuentran al interior de un vector, es necesario recorrer el array o vector, para recorrerlo, se usa casi siempre un ciclo for, en algunos casos mas específicos un ciclo while, pero generalmente el ciclo for es el ideal para esto, dado que conocemos el tamaño del array. La lógica de este procedimiento es la siguiente, el ciclo for comenzara desde cero e ira hasta el tamaño del vector, de modo que la variable de control que generalmente llamamos "i", será la que va a ir variando entre cero y el tamaño del array, de esta forma al poner la i al interior de los corchetes, estaremos accediendo al valor de cada casilla del vector y podremos hacer lo que sea necesario con dicho valor, veamos:

Nota: A veces no es posible determinar con facilidad el tamaño exacto de un vector, pero en C++ existen varias formas de determinar el tamaño de un array o vector fácilmente, aquí explicare un método. Cabe notar que este tamaño es el que ira como tope del ciclo for y sería equivalente a que nosotros mismos, en caso de saber el tamaño del vector, lo pongamos allí, sin embargo, como veremos en otra sección no siempre es posible saber con certeza el tamaño de un vector, es por esto que explico cómo hacerlo.

```
#include "iostream"

using namespace std;

int main()
{
    int edades[] = {1,2,9,8,16,32,9,50,36,20,1,87};
    int limite = (sizeof(edades)/sizeof(edades[0]));
```

```
    for (int i = 0; i < limite; i++)
    {
            cout<<edades[i]<<endl;
    }
}
```

Fig11.

Vamos a ver de forma resumida en qué consiste y que hace cada una de estas líneas

Línea 1:

Tenemos en la primera línea la declaración de un vector que contiene las edades de 12 personas, notemos que entre los corchetes no se puso ningún numero, pues no es necesario, ya que el vector tendrá el tamaño según la cantidad de elementos que declaremos entre las llaves, evidentemente si pusiéramos un 12 entre los corchetes, no habría ningún problema.

Línea 2:

En la segunda línea, tenemos la declaración del límite del ciclo o en otras palabras el tamaño del array. El tamaño de un array se puede calcular de varias formas, aquí lo obtenemos calculando el tamaño del array entero, dividido por el tamaño del primer elemento de dicho array, para mas detalles de esto, verifica la información sobre el operador sizeof.

Línea 3 a 6:

Desde la tercera línea hasta la sexta, tenemos entonces un ciclo for que comienza en cero y termina en el límite (es importante notar que la condición usada es estrictamente menor "<" y no menor o igual "<="), al interior de este ciclo, es donde accedemos a cada uno de los elementos del vector por medio de la sintaxis explicada anteriormente

Línea 5:

La quinta línea es quizá la más vital aunque sin las demás no tendríamos nada. En esta línea, estamos accediendo a cada uno de los elementos del array de edades, un elemento por cada vuelta que da el ciclo, accedemos a cada elemento poniendo entre los corchetes la variable i, que es la que esta cambiando a medida que el ciclo

va girando, así estaremos accediendo a todos los elementos e imprimiéndolos por pantalla

Muy bien, llego el momento de afianzar nuestros conocimientos viendo un ejemplo. Ahora que tenemos claro como declarar un vector en C++, como recorrerlo y como acceder a sus datos, vamos a ver un ejemplo basado en el problema que planteé al inicio de esta sección (el de los libros).

Actividades de aprendizaje y Evaluación

ACTIVIDADES DE EVALUACIÓN

ACTIVIDADES DE EVALUACIÓN DE ESTE MÓDULO SON:

ACTIVIDADES DE EVALUACIÓN DE ESTE OBJETO DE APRENDIZAJE SON:

A. EJERCITACIÓN

1. COMPILAR LOS PROBLEMAS RESUELTOS esta se considera para sus valoración que tenga: Análisis de los problema Resuelto, Corrección de Errores y Sustentación de los mismo) Esta actividad tiene un porcentaje de evaluación cuantitativa como se muestra en la table de actividades.
2. Realizaremos un Parcial Practico para evaluar el aprendizaje adquiridos en la resolución de ejercicios por un periodo de tiempo determinado
3. Para confirmar la internalización de los conceptos fundamentales haremos pruebas cortas para medir la parte básicas del módulo.

B. ESTUDIOS DE CASOS

1. SE Realizará un PROYECTO PRÁCTICO por cada módulo donde se valorará que presenten(Análisis del caso, Resolución del Caso y Sustentación de sus Conclusiones)

C. REPRESENTACIÓN DE CONOCIMIETOS

1. Se presenta una LECTURAS OBLIGATORIAS para complementar el material del módulo donde buscamos que al final de esta cada estudiante pueda diseñar y construir en una herramienta online el Mapas Conceptual

PROBLEMA RESUELTOS

LEA Y ANÁLISIS LOS CASOS RESUELTOS DE LOS ARREGLOS UNIDIMENSIONALES

- Con esta práctica resuelta busca reforzar el material de contenidos proporcionados.
- Compiles cada uno de los casos resueltos en el compilador de su escritorio
- Compiles al menos tres casos en un recurso online y muestre los resultados en la RETROALIMENTACIÓN.
- Muestre la prueba de escritorio de al menos 3 casos .

1. **Que rellene un array con los 100 primeros números enteros y los muestre en pantalla en orden ascendente.**

```
#include <iostream>

#include <stdlib.h>

Using namespace std;


int main(void)

{

        int x,tabla[100];


        for (x=1;x<=100;x++)

        {

     tabla[x]=x;

  }


        for (x=1;x<=100;x++)

        {

Cout <<"\n",tabla[x];

  }


  system("PAUSE");

  return 0;

}
```

2. **Que rellene un array con los 100 primeros números enteros y los muestre en pantalla en orden descendente.**

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main(void)
{
          int x,tabla[100];

          for (x=1;x<=100;x++)
          {
      tabla[x]=x;
   }

          for (x=100;x>=1;x--)
          {
      Cout <<"\n",tabla[x];
   }

   system("PAUSE");
   return 0;
}
```

3. **Que rellene un array con los números primos comprendidos entre 1 y 100 y los muestre en pantalla en orden ascendente.**

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main(void)
{
        int x,cont,z,i,tabla[100];

        i=0;
        for (x=1;x<=100;x++)
        {
      cont=0;
      for (z=1;z<=x;z++)
      {
        if (x%z==0)
        {
          cont++;
        }
      }

      if (cont==2 || z==1 || z==0)
      {
       tabla[i]=x;
       i++;
      }
```

```
   }


         for (x=0;x<i;x++)

         {

      cout<<"\n",tabla[x];

   }


   system("PAUSE");

   return 0;

}
```

Proyecto Práctico

Duración : 2:00

Agrupamiento:1

PROYECTO PRÁCTICO

Desarrolle un programa en C++ utilizando un compilador ONLINE para resolver el siguientes enunciado:

1. Desarrolle un programa que muestre un reporte de las estadísticas de los estudiantes por facultad, por escuela y por carrera de la UNIVERSIDAD de PANAMA.

No sabemos cuántos Facultades hay en la UP, dentro de cada Facultad no sabemos cuántas escuelas hay y dentro de cada Escuela tampoco sabemos la cantidad de carrera. Se le pide que imprima el siguiente Reporte

Universidad de Panamá

REPORTE DE ESTADISTICA

I SEMESTRE 2019

Facultad: FIEC

Escuela	Carrera-1	Carrera-n	TOTAL-E
INFORMATICA	55	35	90
ELECTRONICA	15	40	55

Total de Estudiante por Escuela: xx

Total Estudiante por Facultad:xx

Facultad: FAECO

Escuela	Carrera-1	Carrera-n	TOTAL-E
INFORMATICA	55	35	90
ELECTRONICA	15	40	55

Total de Estudiante por Escuela: xx

Total Estudiante por Facultad:xx

Total de Estudiante de UP:xx

Cuál es la Facultad con mayor cantidad de estudiante: xx

Cuál es la Carrera con Menos Cantidad de Estudiante: xx

Nota: Use funciones por lo menos para cargar la matriz y otra para imprimir la salida.

Lecturas Obligatorias

Lecturas Obligatorias

En esta actividad Cada estudiante debe construir un mapa conceptual referente al tema de la lectura obligatoria que reforzará los conceptos y casos complementarios para que cada uno pueda ampliar sus conocimientos y aprendizajes.

Esta actividad es en equipo de dos estudiantes cada uno, pero debe enviarla por separados.

Presione Aquí Para Descargarla

La evaluación de los aprendizajes es un componente del proceso educativo, a través del cual se observa, recoge y analiza información significativa, respecto de las posibilidades, necesidades y logros de los alumnos, con la finalidad de reflexionar, emitir juicios de valor y tomar decisiones pertinentes y oportunas para el mejoramiento de sus aprendizajes.

EVALUACIÓN

NUM	ACTIVIDAD DE APRENDIZAJE	ACTIVIDAD DE EVALUACIÓN	ESCALA DE EVALUACIÓN	VALORIZACION	%
1	EJERCITACIÓN	COMPILAR PROBLEMA RESUELTO(Análisis de los problema Resuelto, Corrección de Errores y Sustentación de los mismo)	EXCELENTE	95 a 100	5
			BUENO	85 a 95	
			REGULAR	75 a 85	
			DEFICIENTE	Menos de 75	
2		PARCIAL(Resolución de Ejercicios)	EXCELENTE	95 a 100	5
			BUENO	85 a 95	
			REGULAR	75 a 85	
			DEFICIENTE	Menos de 75	
3		QUIZ(Verificación de aprendizaje por preguntas cortas y ejercicios sencillos)			5
4	ESTUDIO DE CASOS	PROYECTO PRÁCTICO(Análisis, Resolución del Casos y Sustentación de sus Conclusiones)	EXCELENTE	95 a 100	5
			BUENO	85 a 95	
			REGULAR	75 a 85	
			DEFICIENTE	Menos de 75	
5	REPRESENTACIÓN DE CONOCIMIENTOS	LECTURAS OBLIGATORIAS(Construcción de Mapas Conceptual)	EXCELENTE	95 a 100	5
			BUENO	85 a 95	
			REGULAR	75 a 85	
			DEFICIENTE	Menos de 75	
TOTAL					**25**

CONCLUSIONES

- En este Objeto de aprendizaje el estudiantes será capaz de identificar que es un arreglos y su ubicación dentro de los tipos de estructura de datos.
- El estudiantes clasificará los diferentes tipos de arreglos, su uso y comprender de qué manera acceden a la memoria.
- Manejará los compiladores de C++ para realizar sus prácticas, proyectos y laboratorios del OA.
- El estudiantes desarrollará las operaciones con arreglos como lectura, recorrido e impresión.

BIBLIOGRAFÍA

- Quetglás, G., Toledo, F., & Cerverón, V. (2002). Estructura de datos. En Fundamentos de Informática y Programación (pp. 171–211). Disponible en http://robotica.uv.es/pub/Libro/PDFs/CAPI5.pdf
- Cruz, D. (s.f.). *Apuntes de la asignatura: Estructura de datos*. México: Tecnológico de Estudios Superiores de Ecatepec. Disponible en http://myslide.es/documents/manual-estructura-de-datos.html
- Frittelli, V., Steffolani, F., Harach, J., Serrano, D., Fernández, J., Scarafia, D., Teicher, R., Bett, G., Tartabini, M. &Strub, A. (s.f). *Archivos Hash: Implementación y aplicaciones*. Universidad Tecnológica Nacional, Facultad Regional Córdoba. Disponible en http://conaiisi.unsl.edu.ar/2013/121-493-1-DR.pdf
- Joyanes, L. y Zahonero, I. (2003). *Programación en C. Metodología, algoritmos y estructura de datos*. Málaga,España: McGraw Hill.
- Loomis, M. E. (1995). *Estructura de datos y organización de archivos*. México: Prentice-Hall Hispanoamericana.

2.OA -DIAGRAMA DE FLUJO

ASIGNACIÓN INDIVIDUAL

DIAGRAMA DE FLUJO

Descripción

En esta actividad busca evaluar el aprendizaje de los estudiantes referentes a los conocimientos adquiridos de las técnicas de programación partiendo del diagrama de flujo para lograr obtener el seudocódigo o algoritmo en leguaje natural usando PSI para realizarlo. Adicional deben usar cualquier compilador ONLINE para compilar dicho problema en c++.

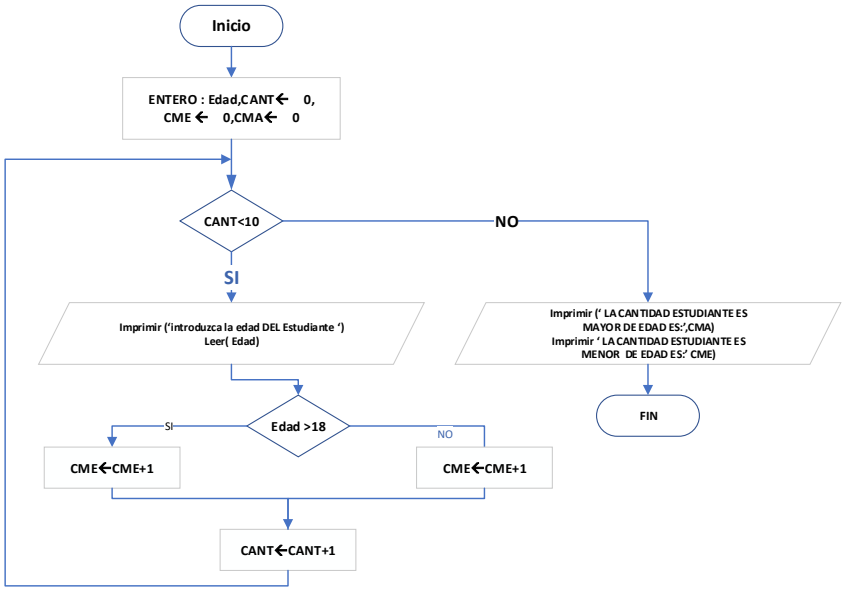
OBJETIVO:

- Conocer la sintaxis del lenguaje c++ transformando los diagramas de flujo a seudocódigo y a programas en c++

ACTIVIDAD

Desarrolle los siguientes ejemplos

1. Diseñe el seudocódigo y el programe en c++ que lea la edad de diez estudiantes del cuso de lógica de programación e imprima cuantos son estudiantes son mayores de 18 años y cuantos son menores de 18 años. Utilice la Estructura Repetitiva mientras.



Nota: Envié los dicha actividad al correo del tutor como:
Asunto: ACTIVIDAD 1 INDIVIDUAL
Nombre Archivo: ApellidoNombre.doc
Adicional el C++ debe compilarlo en un compilador online y enviar el URL.

2. OA DIAGRAMA DE CASOS DE USOS

DIAGRAMA DE CASOS DE USOS

El diagrama de casos de uso es uno de los diagramas incluidos en UML 2.5, estando este clasificado dentro del grupo de **diagramas de comportamiento**. Es, con total seguridad, el diagrama más conocido y es utilizado para representar los actores externos que interactúan con el sistema de información y a través de que funcionalidades (casos de uso o requisitos funcionales) se relacionan. Dicho de otra manera, muestra de manera visual las distintas funciones que puede realizar un usuario (más bien un tipo de usuario) de un Sistema de Información.

ACTIVIDAD

Una vez desarrollado el TEMA de este MODULO Casos de Usos, cada estudiantes debe:
Desarrolle tres casos de uso de un sistema de información de Administración y gestión de una biblioteca tomando en cuenta tres requerimientos diferentes. Identifique cada ACTOR, CASOS DE USO. Utilice la notación dada en clases para representar cada Diagrama de Casos de Uso.

Entrega:

Nota: Envié los dicha actividad al correo del tutor como:

Asunto: ACTIVIDADINDIVIDUAL_NOMBREAPELLIDO_CDU_A3

Nombre Archivo: NOMBREAPELLIDO_CDU_A3.doc

Use las siguientes herramientas online:

https://www.lucidchart.com

O descargue las siguientes herramientas:

- https://sourceforge.net/projects/staruml/
- http://argouml.tigris.org/