PROGRAMACION II - 2025

DANIEL KLOSTER

MAXIMILIANO WENNER

ANGEL SIMON

BRIAN LARA

VERONICA CARBONARI

CARACTERÍSTICAS DE LA MATERIA

- CLASES TEORICA - PRÁCTICAS
- TRABAJO EN EL AULA (O EN LA CASA) SOBRE LA COMPUTADORA O PAPEL Y LÁPIZ
- SE DESARROLLA EN DOS ENCUENTROS SEMANALES
- EL LENGUAJE A UTILIZAR ES C/C++.
- ENTORNO DE TRABAJO: AULA VIRTUAL, CODEBLOCKS, CUADERNO DE CLASE.
- SE RECOMIENDA EL TRABAJO EN PEQUEÑOS GRUPOS. FAVORECE EL APRENDIZAJE

CONTENIDO

- **PARTE 1: CONTINUAMOS CON PROGRAMACIÓN ESTRUCTURADA**
  - TEMAS:
    - REPASO DE VARIABLES Y FUNCIONES. TRABAJO CON VECTORES, CADENAS.
    - PUNTEROS, MATRICES , REGISTROS
    - ASIGNACIÓN DINÁMICA DE MEMORIA

CONTENIDO

- **PARTE 2: EMPEZAMOS A TRABAJAR CON PROGRAMACIÓN ORIENTADA A OBJETOS (P.O.O)**
  - TEMAS:
    - DEFINICIONES: CLASES, OBJETOS. BASES PARA EL DISEÑO DE CLASES. CODIFICACIÓN EN C++.
    - MECANISMOS PARA EL DISEÑO DE CLASES: HERENCIA Y COMPOSICIÓN
    - SOBRECARGA DE OPERADORES Y FUNCIONES
    - ASIGNACIÓN DINÁMICA DE MEMORIA EN C++
  - **ARCHIVOS:** DEFINICIÓN. UTILIDAD. TIPOS DE ARCHIVOS. OPERACIONES CON ARCHIVOS
  - INTRODUCCIÓN A LOS SERVICIOS WEB

CONTENIDO

- **EVALUACIONES:**

  - **PRIMER EVALUACIÓN PARCIAL 1 (P1):** CONSISTE EN LA RESOLUCIÓN DE UN CONJUNTO DE ENUNCIADOS PROBLEMA BASADOS EN ESTRUCTURAS DE DATOS AVANZADAS, ARCHIVOS Y CONCEPTOS BÁSICOS DE POO. SE ENTREGA A CADA ESTUDIANTE EN LA FECHA DEL PARCIAL.

  - **SEGUNDA EVALUACIÓN TRABAJO PRÁCTICO INTEGRADOR (IN):** CONSISTE EN EL DESARROLLO DE UN SISTEMA FUNCIONAL DE GESTIÓN DE ARCHIVOS (O UN JUEGO), QUE SE IRÁ DESARROLLANDO DE MANERA GRUPAL DURANTE LAS CLASES SIGUIENTES A LA REALIZACIÓN DEL PRIMER PARCIAL. CONSTA DE LAS SIGUIENTES ETAPAS
    - 1° ETAPA: SELECCIÓN DEL GRUPO (MÁXIMO CUATRO INTEGRANTES POR GRUPO).
    - 2° ETAPA: SELECCION DEL PROGRAMA A DESARROLLAR Y PRESENTACIÓN DEL INFORME DE LOS ARCHIVOS NECESARIOS.
    - 3° ETAPA: INFORME GENERAL DESCRIPTIVO DEL SISTEMA A DESARROLLAR (NOMBRE, ALCANCE, ARCHIVOS Y CLASES).
    - 4° ETAPA: PRESENTACIÓN GRUPAL Y DEFENSA INDIVIDUAL DEL PROGRAMA

    - LAS ETAPAS 1 Y 2 SE CUMPLIMENTAN INTEGRAMENTE EN EL AULA VIRTUAL.
    - LAS ETAPAS 3 Y 4 REQUIEREN DE LA PRESENTACIÓN DE LOS INTEGRANTES DEL GRUPO Y LA DEFENSA DE LO PRESENTADO.

    - LA NO PRESENTACIÓN EN ALGUNA ETAPA (GRUPO COMPLETO O ALGÚN INTEGRANTE) RESTA UN PUNTO EN LA CALIFICACIÓN FINAL

CONTENIDO

- **EVALUACIONES:**
  - **CADA EVALUACIÓN TIENE UNA INSTANCIA DE RECUPERACIÓN, DE SIMILARES CARACTERÍSTICAS A LAS ORIGINALES DE LA EVALUACIÓN.**
  - **CRITERIOS DE REGULARIZACIÓN Y APROBACIÓN DIRECTA**
    - **PARA REGULARIZAR LA MATERIA ES NECESARIO OBTENER UN 6 O MÁS EN AMBAS CALIFICACIONES (P1 E IN).**
    - **PARA LOGRAR LA APROBACIÓN DIRECTA DE LA MATERIA ES NECESARIO OBTENER AL MENOS UN 6 EN P1 Y UN 8 O MÁS EN IN. AL LOGRAR LA APROBACIÓN DIRECTA SE ASEGURAN UNA CALIFICACIÓN DE APROBACIÓN EN EL EXAMEN FINAL QUE CONSISTIRÁ EN EL PROMEDIO DE AMBAS CALIFICACIONES.**
  - EN EL CONTRATO DIDÁCTICO Y EL DOCUMENTO PREGUNTAS FRECUENTES PUEDEN ENCONTRAR MÁS DETALLES SOBRE ESTOS TEMAS
  - EN EL CRONOGRAMA ESTÁN LAS FECHAS DE LAS EVALUACIONES Y LOS RECUPERATORIOS

REPASO

- VARIABLES
- FUNCIONES

VARIABLES

- CLASIFICACION:
  - TIPO (QUE PUEDE ALMACENAR): int, float, char, bool y direcciones de tipo definido. Todos los lenguajes tienen tipos de datos definidos (se los suele llamar primitivos o integrados), y proveen mecanismos para que el usuario pueda definir nuevos tipo. En nuestro caso vamos a crear nuevos tipos de datos con struct y clases.
  - CANTIDAD DE ELEMENTOS QUE CONTIENE: VARIABLES SIMPLES, VECTORES, MATRICES
  - ALCANCE: LOCALES Y GLOBALES

VARIABLES COMPUESTAS: VECTORES Y MATRICES

- SE LAS SUELE DENOMINAR ARRAY (A VECES EN LA TRADUCCIÓN ESPAÑOLA ARREGLOS)
- PUEDEN TENER UNA DIMENSIÓN (VECTORES) O VARIAS (MATRICES)
- EN TODOS LOS CASOS SOLO SE PUEDEN UTILIZAR SI SE CONOCE PREVIAMENTE LA CANTIDAD DE ELEMENTOS QUE CONTENDRÁN. NUNCA SE DEBE DECLARAR UN ARRAY CON UNA VARIABLE PARA DIMENSIONARLO (int a=5; int vec[a];)
- PUEDEN SER DE CUALQUIERA DE LOS TIPOS DE DATOS CONOCIDOS (O QUE SE CONOCERÁN MÁS ADELANTE)

ARRAYS. USO

- SON MUY ÚTILES CUANDO ES NECESARIO TRABAJAR DE MANERA CONJUNTA UNA CANTIDAD DEFINIDA DE DATOS DE UN MISMO TIPO. COMO SE DIJO, ESA CANTIDAD DEBE SER CONOCIDA ANTES DE SU DECLARACIÓN (ESTA LIMITACIÓN SE ELIMINA AL TRABAJAR CON ASIGNACIÓN DINÁMICA DE MEMORIA)

- VECTOR: VARIABLE QUE ALMACENA BAJO UN MISMO NOMBRE DE VARIABLE UN CONJUNTO DE DATOS DEL MISMO TIPO. CADA DATO EN PARTICULAR ES ACCEDIDO MEDIANTE EL NOMBRE DEL VECTOR Y UN SUBINDICE ENTRE CORCHETES. EL PRIMERO DE ELLOS SE ACCEDE CON EL NÚMERO 0. EJEMPLO: vec[0]=8;
- EL NOMBRE DE UN VECTOR ES UN PUNTERO QUE CONTIENE LA DIRECCIÓN DEL ELEMENTO CERO

VECTORES DE CARACTERES

- EN C/C++ LOS VECTORES DE CARACTERES SON UTILIZADOS PARA ALMACENAR CADENAS DE CARACTERES, ESTOS ES, PALABRAS O CONTENIDO ALFANUMERICO, Y POR ESO TIENEN UN TRATAMIENTO ESPECIAL DISTINTO DEL RESTO DE LOS VECTORES (POR EJEMPLO SE PUEDE HACER UN cin>> o un cout<< SOBRE EL NOMBRE DEL VECTOR).
- EL TRATAMIENTO ESPECIAL CONSISTE EN ASIGNARLE UN CARÁCTER ADICIONAL EN EL INGRESO (‘\0’) PARA INDICAR EL FIN DE LA CADENA.

- SI BIEN SE PUEDE TRABAJAR CARACTER A CARACTER (UTILIZANDO CADA POSICIÓN DEL VECTOR) PARA UTILIZAR CADENAS SE UTILIZAN LAS FUNCIONES DE LA LIBRERÍA CSTRING.
- LAS MÁS COMUNES SON:
    - STRCPY(): PARA COPIAR UNA CADENA EN OTRA. SE USA EN LUGAR DEL OPERADOR DE ASIGNACIÓN
    - STRCMP(): SE USA PARA COMPARAR UNA CADENA CON OTRA (NO SE PUEDEN USAR DIRECTAMENTE LOS OPERADORES RELACIONALES)

MATRICES

- SE UTILIZAN CUANDO ES NECESARIO TRABAJAR CON UN CONJUNTO DE DATOS DE UN MISMO TIPO A LOS QUE SE PUEDE DEFINIR COMO PERTENECIENTE A ALGUNA DIMENSIÓN DE UN CONJUNTO DE DIMENSIONES CONOCIDA PREVIAMENTE (DOS Ó MÁS)
- CADA DIMENSIÓN DEBE CONOCERSE PREVIAMENTE A SU DECLARACIÓN. AL IGUAL QUE PARA LOS VECTORES, NO SE PUEDEN USAR VARIABLES.
- PUEDEN SER DE CUALQUIER TIPO DE DATO CONOCIDO O A CONOCER.
- EL NOMBRE DE UNA MATRIZ ES LA DIRECCIÓN (UN PUNTERO) A LA PRIMERA FILA DE ESA MATRIZ. ES DISTINTO AL NOMBRE DE UN VECTOR (PUNTERO AL PRIMER ELEMENTO)

ARRAYS. FUNCIONALIDAD

- DECLARACION:

  ```
  int v[10];

  float m[6][8];
  ```

  - SE DEFINE LA CANTIDAD QUE SE NECESITA
- USO: DESDE 0 A LA CANTIDAD DEFINIDA -1

  ```
  v[0]=56; v[9]=14;
  ```
- PARA PASAR UN VECTOR O UNA MATRIZ ENTERA COMO PARAMETRO: SE PONE EL NOMBRE DE LA VECTOR O LA MATRIZ

  ```
  ponerCero(v,10);

  mostrar(m);
  ```
- PARA RECIBIR EN UNA FUNCION UN VECTOR O UNA MATRIZ: SE UTILIZA EL PUNTERO CORRESPONDIENTE, O LA NOTACIÓN EXPLICITA void ponerCero(int *v) ó void ponerCero(int v[]) ó void ponerCero(int v[10]

FUNCIONES

- DEFINICION:

fragmento de código que resuelve un problema en particular. Tiene el siguiente formato:

```
valor_devuelto nombreFuncion(tipo nombreParametro1, tipo nombreParametro2, .., tipo nombreParametroN){
        ///desarrollo del código de la función
        }
```

valor devuelto: tipo de dato que devuelve la función mediante la instrucción return. Puede devolver SOLO UNO. Si no devuelve nada se pone void.

nombreFuncion: el nombre de la función lo establece quien lo crea. Sigue las mismas reglas que los nombres de variables

tipo nombreParametro1, 2, etc: declaración de los parámetros que donde se van a escribir los valores que se le envían a la función. Para cada uno hay que poner el tipo y el nombre. Si no recibe nada se puede poner void o dejar vacío.

ENTRE LAS LLAVES SE DESARROLLA EL CÓDIGO ESPECÍFICO DE LA FUNCIÓN.

FUNCIONES

- DECLARACIÓN DE UNA FUNCIÓN:

Como toda elemento del código, para que el compilador reconozca a la función tiene que estar declarada antes de ser utilizada. Esto se hace mediante el prototipo que tiene el siguiente formato:

valor_devuelto nombreFuncion(tipo nombreParametro1, tipo nombreParametro2, ., tipo nombreParametroN) ;

luego de su declaración puede hacerse el desarrollo completo de la función en cualquier otra parte del programa. Si el desarrollo se hace de manera completa antes de ser utilizada no es necesario el prototipo.

Se recomienda hacer el prototipo para organizar de manera más clara el código.

FUNCIONES

- **DESARROLLO DE UNA FUNCIÓN:**

  El desarrollo de una función es el código completo. Contiene las líneas que se ejecutarán cuando la función sea llamada.

- **LLAMADA DE UNA FUNCIÓN:**

  Cuando en main() u otra función encontramos el nombre de una función, se dice que se llama a la función. Si la llamada es correcta (existe la función y se le envían los parámetros necesarios) el programa continuará su ejecución DENTRO DE LA FUNCIÓN LLAMADA.

  En la llamada a una función no se indica el valor devuelto, ni el tipo y nombre de los parámetros. Si la función devuelve valor, éste puede ser almacenado en una variable o usado de manera directa; si la función espera parámetros es necesario poner en la llamada los valores que correspondan.

RESUMEN VECTOR

- PUEDE SER DE CUALQUIERA DE LOS TIPO DE DATOS CONOCIDOS
- SE LOS DECLARA CON UNA CONSTANTE QUE INDIQUE LA CANTIDAD DE ELEMENTOS QUE SE NECESITAN
- SE UTILIZAN DESDE LA POSICIÓN 0 A LA POSICIÓN CANTIDAD DE ELEMENTOS -1
- CUANDO SE LOS PASA COMO PARÁMETRO DE UNA FUNCIÓN SE COLOCA EL NOMBRE SOLO
- EL NOMBRE DE UN VECTOR ES UN PUNTERO CONSTANTE QUE CONTIENE LA DIRECCIÓN DEL PRIMER ELEMENTO DEL VECTOR, POR LO TANTO TIENE QUE SER RECIBIDO EN LA FUNCIÓN CON UN PUNTERO. SE LO PUEDE REPRESENTAR INDISTINTAMENTE DE LAS SIGUIENTES FORMAS:

  tipo *nombre

  tipo nombre[ ] con los corchetes vacíos

  tipo nombre[25] con la dimensión con la que fue declarado
- LAS DOS PRIMERAS FORMAS SON GENÉRICAS. LA ÚLTIMA ES SÓLO PARA UN VECTOR DE ESA CANTIDAD DEFINIDAD DE COMPONENTES

RESUMEN VECTOR

EJEMPLOS:

- int vec1[15]:
- char vec2[30];
- int *v[15]//ESTE ES UN VECTOR DE PUNTEROS A ENTEROS

SUPONIENDO UNA FUNCIÓN DE NOMBRE func1() que recibe un vector de enteros de 10 elementos y no devuelve nada , su cabecera podría escribirse de las siguientes maneras:

```
void func1(int *v)
void func1(int v[ ])
void func1(int v[10])
```

RESUMEN MATRIZ

- PUEDE SER DE CUALQUIERA DE LOS TIPO DE DATOS CONOCIDOS
- PUEDE TENER DOS Ó MÁS DIMENSIONES, CADA UNA DE ELLAS TENDRÁ UN PAR DE CORCHETES QUE ACEPTARÁ UN VALOR ENTERO PARA INDICAR A CUAL DE LOS ELEMENTOS DE ESA DIMENSIÓN SE REFIERE
- SE LAS DECLARA CON UNA CONSTANTE QUE INDICA LA CANTIDAD DE ELEMENTOS QUE SE NECESITAN EN CADA UNA DE SUS DIMENSIONES
- SE UTILIZAN DESDE LA POSICIÓN 0 A LA POSICIÓN CANTIDAD DE ELEMENTOS -1
- CUANDO SE LOS PASA COMO PARÁMETRO DE UNA FUNCIÓN SE COLOCA EL NOMBRE SOLO
- EL NOMBRE DE UNA MATRIZ DE DOS DIMENSIONES ES UN PUNTERO CONSTANTE QUE CONTIENE LA DIRECCIÓN DE LA PRIMERA FILA DE ESA MATRIZ, POR LO TANTO TIENE QUE SER RECIBIDO EN LA FUNCIÓN CON UN PUNTERO DE ESAS CARACTERÍSTICAS. SE LO PUEDE REPRESENTAR INDISTINTAMENTE DE LAS SIGUIENTES FORMAS:

  tipo (*nombre)[CANTIDAD DE COLUMNAS]
  tipo nombre[ ][25] con los corchetes que indican las filas vacíos
  tipo nombre[10][25] con las dos dimensiones con las que fue declarada
- NINGUNA DE LAS FORMA VISTAS SON GENÉRICAS PORQUE EXIGEN QUE SE DECLARE AL MENOS LA CANTIDAD DE COLUMNAS

RESUMEN MATRIZ

EJEMPLOS:

- int mat1[10][15]:
- char mat2[30][30];
- int *v[15][10]//ESTA ES UNA MATRIZ DE PUNTEROS A ENTEROS

SUPONIENDO UNA FUNCIÓN DE NOMBRE func1() que recibe una matriz de floats de 10 filas y 20 columnas y no devuelve nada , su cabecera podría escribirse de las siguientes maneras:

```
void func1(int (*m)[20])
void func1(int m[][20])

void func1(int m[10][20])
```

RESUMEN VECTOR DE CARACTERES (CADENAS)

- SE LOS UTILIZA EN C PARA REPRESENTAR PALABRAS, O CONJUNTO DE CARACTERES QUE SE TRABAJAN COMO UNA UNIDAD Y QUE INCLUYEN LETRAS Y NÚMEROS
- SE LOS DECLARA CON UNA CONSTANTE QUE INDIQUE LA CANTIDAD DE ELEMENTOS QUE SE NECESITAN MÁS 1, QUE SE RESERVA PARA EL CARÁCTER ESPECIAL '\0', TAMBIÉN CONOCIDO COMO TERMINADOR
- SE LOS PUEDE UTILIZAR POSICIÓN POR POSICIÓN, PERO EN LA LIBRERÍA STRING.H (O CSTRING) HAY UN CONJUNTO DE FUNCIONES PARA SU MANIPULACIÓN. LAS MÁS USUALES SON:
- strcpy(char *cad1, const char *cad2) copia la cadena cad2 en cad1
- int strcmp(const char *cad1, const char *cad2) permite comparar cadenas.
- int strlen(char *cad) devuelve la cantidad de caracteres de la cadena cad.

- LO QUE DETERMINA EL COMPORTAMIENTO DE UN VECTOR DE CHAR O CADENA ES EL CARÁCTER NULO. CUANDO SE INGRESA LA CADENA, AL PRESIONAR ENTER, SE LE INSERTA EL '\0' DE MANERA AUTOMÁTICA.
- CUANDO SE LEE UNA CADENA SE LO HACE DESDE LA DIRECCIÓN DEL ELEMENTO 0 (ESTÁ CONTENIDA EN EL NOMBRE DEL VECTOR) HASTA QUE SE ENCUENTRE EL CARÁCTER NULO.
- PARA TRABAJAR UN CONJUNTO DE CADENAS A LA VEZ SE PUEDE UTILIZAR UNA MATRIZ DE CARACTERES. CADA FILA PODRÁ CONTENER UNA CADENA. PARA ACCEDER A CADA UNA DE LAS CADENAS BASTA CON HACER REFERENCIA A LA PRIMERA DIMENSIÓN DE LA MATRIZ (LA FILA)

EJERCICIO

LAS AUTORIDADES DE LA CARRERA TUP DE LA UTN ESTÁN REALIZANDO UN ANÁLISIS DE LOS CURSOS VIRTUALES DE LAS DISTINTAS MATERIAS. POR CADA UNA DE LAS 20 MATERIAS DE LA CARRERA TIENE LA SIGUIENTE INFORMACIÓN

- NÚMERO DE MATERIA (ENTRE 1 Y 20), NOMBRE, CANTIDAD DE ALUMNOS INSCRIPTOS, CANTIDAD DE PROFESORES

ADEMÁS POR CADA INGRESO DE LOS ESTUDIANTES AL AULA VIRTUAL SE REGISTRA LO SIGUIENTE:

- LEGAJO, FECHA DE ACCESO (DÍA Y MES), NÚMERO DE LA MATERIA A LA QUE INGRESO, CANTIDAD DE HORAS (NÚMERO REAL)

EL FIN DE LOS DATOS SE INDICA CON UN NÚMERO DE LEGAJO IGUAL A 0.

SE QUIERE RESPONDER LAS SIGUIENTES PREGUNTAS:

a) LAS MATERIAS QUE NO TUVIERON ACCESO DE ALUMNOS NUNCA

b) LA MATERIA QUE MÁS CANTIDAD DE HORAS REGISTRO DE ACCESO DE ALUMNOS

c) POR CADA MATERIA Y DÍA DE MARZO, LA CANTIDAD DE ACCESOS DE ALUMNOS A LAS AULAS VIRTUALES.

- HACER UN PROGRAMA EN EL MARCO DE UN PROYECTO DE CODEBLOCK CON UN MENÚ CON OPCIONES PARA CARGAR LOS DATOS, MOSTRAR CADA PUNTO Y SALIR DEL PROGRAMA.

EJERCICIO

AGREGAR LAS SIGUIENTES PREGUNTAS AL EJERCICIO:

d) LA CANTIDAD DE MATERIAS CON ACCESOS

e) POR CADA MATERIA LA CANTIDAD DE ALUMNOS POR PROFESOR

f) POR CADA MATERIA Y DÍA DE MARZO, LA CANTIDAD DE HORAS DE ACCESOS DE ALUMNOS A LAS AULAS VIRTUALES.