Universidad de Los Andes
Escuela de Ingeniería de Sistemas
Departamento de Computación

PROGRAMACIÓN 2

Clase 01

Junior Altamiranda

altamira@ula.ve

AGENDA

- **El Curso**
- **Evaluación**

- **Materia**
- Repaso de PR1 con C++
- Apuntadores

Datos del Curso

Horario:

- **Martes** Salon 2S09 hora: 8 a 10 am
  Viernes Salon 2O08 hora: 8 a 10 am

- **Laboratorio**: Salon 2S09 miercoles 10 a 12 am

- **Profesor Junior Altamiranda**
  CEMISID 3 er piso

- **Clases: Dropbox**

Evaluación

Parcial 1: (19/02/2013) 15%
Parcial 2: (12/03/2013) 15%
Parcial 3: (09/04/2013) 15%
Parcial 4: (30/04/2013) 15 %
Proyecto: (03/05/2013) 15%
Diferido: (07/05/2013)
Laboratorio: 15%
Tareas 10%

Proyecto

Grupo: 4 personas

Sistema de gestión de inventario

Observaciones

Para aprobar la materia se requiere un porcentaje de asistencia superior al 75%

Para aprobar el Laboratorio se requiere un porcentaje de asistencia superior al 75% y aprobar las prácticas => 10 ptos

Ambientes de programación

- **Sistema Operativo Linux (Debian)**
- Compilador g++ (GNU Compiler for OOP)
- Editores vi, emacs, gedit

- **Pueden utilizar su laptop (Con Windows o Linux)**
- Pero no en los exámenes

¿Qué veremos en PR 2?

Creación dinámica de memoria

- Repaso de apuntadores
- Repaso de registros
- Operadores new y delete

Tipos de datos abstractos y Programación Orientada a Objetos

- Definiciones
- Constructos (Herencia, polimorfismo, encadenamiento dinámico, etc.)
- TDAs Pila y Cadena
- Diagrama de clases (y Diagrama de Casos de Uso)
- Ejemplos

Archivos

- Secuenciales
- De acceso directo (Aleatorio)

Parcial 1

Punteros

Tipos de datos abstractos
- Definición

Programación Orientada a Objetos
- Definición de la POO
- Definición de clases
- Definición de objetos
- Definición de identidad de objetos

Parcial 2

Programación Orientada a Objetos

- Definición de herencia, clasificación
- Definición de encadenamiento dinámico
- Definición de clases paramétricas

Tipos de Dato Abstracto (TDA)

- Clasificació n y especificación
- EL TDA Cadena

Especificación de Programas Orientados a Objetos

Parcial 3

Almacenamiento secundario:

- Conceptos básicos
- Archivos secuenciales. Buscar, Agregar, Eliminar, Modificar información

Archivos tipo Texto

Parcial 4

Archivos Acceso Directo.

- **Transformación clave dirección – Archivos Hash**
-
- **Archivos de acceso secuencial indexado**
-
- **Creación de índices densos y no densos**

Orígenes de C y C++

- C surgió de otros dos lenguaje: BCPL y B
- BCPL fue desarrollado en 1967 por Martin Richard
- Ken Thompson modeló muchas de las características de B, las cuales inspiraron al lenguaje C
- El lenguaje C fue una evolución de lenguaje B llevada a cabo por Dennis Ritchie en 1972.

- Bjarne Stroustrup desarrolló C++, que es una extensión de C, a principios de los 80.
- C++ es una evolución de C. Su principal característica es que proporciona capacidades para la Programación Orientada a Objetos.

Pasos al compilar y ejecutar

1. **Edición**: Se crea el programa y se almacena en un archivo de texto, generalmente con extensión .cpp, .c, .C
2. **Preprocesamiento**: Inclusión de archivos, se quitan los comentarios, reemplazos de texto.
3. **Compilación**: Se convierte el código en lenguaje de alto nivel a leng. de máquina.
4. **Enlace**: Se vincula el código con las funciones de bibliotecas que requiere. Se genera un ejecutable.
5. **Carga**: Se almacena en memoria el programa.
6. **Ejecución**: El CPU procesa las instrucciones una a una según indique el contador de programa.

Hola Mundo de C++

(Compilador **g++**)

```
#include <iostream>
using namespace std;

int main() {
        cout << “Hola mundo!” << endl;
        return 0;
}
```

Comentarios del Hola Mundo C++

iostream: Biblioteca de funciones para flujos de entrada y salida

cin: Objeto de flujo de entrada estándar

cout: Objeto de flujo de salida estándar

endl: Manipulador de flujo. Envía un salto de línea y descarga el buffer de salida

<<: Operador de inserción de flujo (Concatena)

>>: Operador de extracción de flujo

REPASO

Variables

- Nombre dado a una localidad de memoria
- Posee: Nombre, dirección, tipo, tamaño y valor
- Al asignarle un valor a una variable, se "destruye" el valor anterior: Se pierde.
- Al leer el valor de una variable, este valor se mantiene en la localidad de memoria: No se pierde

Tipos de datos

Enteros:
int
short int
long int
unsigned short int
unsigned int
unsigned long int

Punto Flotante:
float (IEEE754 Simple)
double (IEEE754 Normal)
long double (IEEE754 Extendido)

Carácter:
char

Booleano:
bool

Operadores aritméticos

Nombre	Símbolo	Ejemplo
Suma	+	x = 7 + 5; // x = 12
Resta	-	x = 7 - 5; // x = 2
Multiplicación	*	x = 7 * 5; // x = 35
División	/	x = 7 / 5; // x = 1
Residuo	%	x = 7 % 5; // x = 2

Operadores lógicos

Nombre	Símbolo	Ejemplo
Igual a	==	x == y
Diferente a	!=	x != y
Negación	!	!x
Mayor	>	x > y
Menor	<	x < y
Mayor o igual a	>=	x >= y
Menor o igual a	<=	x <= y

Identificadores válidos

- Cualquier combinación de letras, dígitos y _ que no comiencen con un dígito.
- Sensibles a mayúsculas y minúsculas
- Sin caracteres especiales: tildes, acentos, asteriscos, etc.
- Se recomienda usar identificadores descriptivos, pero de 31 caracteres o menos.

RECOMENDACIONES

- Indentar el texto. No usar tabulaciones sino espacios (2 a 4).
- Conocer las bibliotecas (reutilizar componentes)
- Mantener, lo más sencillo posible, la lógica y legibilidad de la programación: A menos que logre una ventaja significativa
- Documentar (// y /* */) todo lo que desee, el programa que se esté realizando: ¡Es gratis!
- Revisar detalladamente los errores arrojados por el compilador
- Colocar espacio después de las comas y antes y después de los símbolos. Colocar líneas en blanco para dividir las partes del programa
- Usar nombres descriptivos y prefijos para las variables (que indiquen el tipo de dato). Pero de 31 caracteres o menos
- No colocar más de una instrucción por línea.

Ejercicio en C++

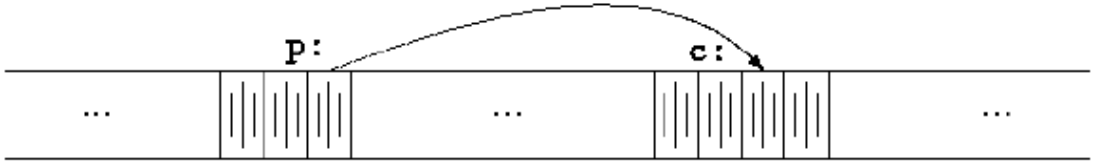
Realizar un programa en C++ que solicite el nombre al usuario y le diga a este cuántas letras tiene su nombre.

MATERIA

Apuntadores (Punteros)

- Apuntador: Variable que guarda una dirección de memoria de otra variable.
- Ventaja general: Código más compacto y eficiente.
- Se pueden tener apuntadores a cualquier tipo de variable
- Usos primordiales:
- Reservar y liberar memoria dinámicamente (Eficiencia)
- Manejo de vectores o arreglos (aritmética de punteros)
- Pase de parámetros por referencia

Apuntadores



Operadores:

* (Indirección) Retorna en contenido:
  Sintaxis: *<variable_apuntador>
& (Dirección) Retorna la dirección:
  Sintaxis: &<variable>

Ejemplos con apuntadores

```
int x = 23;
int *p = &x;

*p=35;
```

Errores:

```
p=35;
*x=35;
```

Apuntadores y arreglos

- Al declarar un arreglo, el identificador de este, es un apuntador.
- La función debe declarar un parámetro apuntador con el tipo de datos de los elementos del vector.
- Por ejemplo: char nombre[20]

float sueldo[50];

Nombre y sueldo, tienen como valor la primera dirección de memoria del vector.
Nombre y sueldo equivalen a &nombre[0] y a &sueldo[0]

Las funciones declararán:

char * nombres y float * sueldos

Aritmética de punteros

- Un apuntador, apunta a una dirección de memoria de una variable.
- El lenguaje C, permite sumar y restar cantidades para movernos en el espacio de direcciones de memoria.
- Ejemplo:

```
float sueldo[30];
float *p = sueldo+4;
```

p contendrá la dirección del quinto (índice 4) elemento float del vector

```
*(p+2) = 9.6;  //equivale a sueldo[6] = 9.6;
```

Asigna 9.6 al séptimo elemento del vector sueldo.

Parámetros por referencia

- Necesidad:
- Se desea modificar el valor de los parámetros.
- Se desea pasar como parámetros estructuras complejas como arreglos.
- Retornar más de un valor con una función.
- Se le pasa a la función la dirección de memoria donde se encuentra el dato: No se le pasa el valor.
- Para las variables básicas se utiliza el &
- Para los vectores, no se utiliza operador.

Ejercicio

Realizar un programa en C++ que contenga una función que se encarga de leer el Nombre, cédula y sueldo de un empleado. El programa principal debe hacer uso de esta función. Es decir, los datos (nombre, cédula y sueldo) son pasados al programa principal.
No se pueden utilizar variables globales.