INSTITUTO SUPERIOR TECNOLÓGICO

JAPÓN

Amor al conocimiento

GUÍA METODOLÓGICA

LENGUAJE Y PROGRAMACIÓN

DESARROLLO DE SOFTWARE

COMPILADOR: MGS. DIANA MONCAYO

2019

INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN
GUIA DE APRENDIZAJE

1. IDENTIFICACIÓN DE

Nombre de la Asignatura: LENGUAJE Y PROGRAMACION	**Componentes del Aprendizaje**	COGNOSCITIVOS

Resultado del Aprendizaje:

- Comprende el diseño de algoritmos.
- Comprende la programación estructurada y una introducción a la programación al leguaje de desarrollo C++
- Aplica los conocimientos relacionados a las aplicaciones que se pueden usar para aplicar un lenguaje de programación.
- Diseña, analiza y usan la creatividad realizando programas que solventen una necesidad.
- Reconoce la necesidad de usar una herramienta de programación para su aprendizaje y desarrollo profesional.
- Utiliza las técnicas, estrategias y herramientas de la ingeniería moderna necesarias para la práctica de la misma.
- Aplica los conocimientos relacionados a los lenguajes de programación, modelado de sistemas de información, construcción de software de calidad y administración de recursos tecnológicos.
- Desarrolla la capacidad de razonamiento lógico (analizar y aplicar) a través de la práctica en la construcción de algoritmos y su codificación en el Lenguaje de Programación C++.
- Fortalece las habilidades en el Proceso de desarrollo (construcción o adaptación) de sistemas informáticos para las empresas. Conociendo las Metodologías y Técnicas en la Creación, Desarrollo de Algoritmos y su Codificación en un lenguaje de programación determinado.
- Construye algoritmos eficaces para la solución de problemas computacionales básicos

COMPETENCIAS

- Conoce y elabora un programa en un lenguaje de programación, analiza y descubre los errores de ejecución a fin de solventarlos.
- Conocer las herramientas, espacios y recursos del Aula Virtual.
- Adquiere habilidades de trabajo en equipo en un entorno virtual de aprendizaje.
- Obtiene habilidades de análisis, tratamiento, interpretación, elaboración y estructuración de la información digital.
- Monitorizar y facilitar analíticamente las actividades relacionadas con los conocimientos.
- Analiza problemas para diseñar soluciones de aplicaciones con software de programación en C++
- Diseña aplicaciones de software de programación en C++
- Implementa algoritmos mediante software de programación de Lenguaje C++.

OBJETIVOS

- Capacitar a los alumnos en el uso de herramientas y aplicaciones usando el lenguaje de programación de programación en el entorno de NetBeans C++, que se aplicarán en el laboratorio de cómputo.
- Motivar al estudiante en la utilización de las aplicaciones para la ejecución de programas.
- Al finalizar el ciclo el estudiante deberá estar en la capacidad de poder manipular la codificación utilizada en el lenguaje de programación y aplicativos en forma práctica en el desarrollo de asignaturas en su formación profesional.
- Impartir al alumno el desarrollo de clases netamente prácticas con materiales visuales prácticos que se podrán comprobar y practicar en computadoras fuera de hora de clases.
- Apoyar al estudiante a desarrollar los diversos problemas que se presentan en los ejercicios prácticos, realizando ejercicios participativos que denoten sus errores a fin de solventarlos

INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN
GUIA DE APRENDIZAJE

Docente de Implementación: Msc. Diana Moncayo				
		Duración: 25 horas		
Unidades	**Competencia**	**Resultados de Aprendizaje**	**Actividades**	**Tiempo de Ejecución**
Unidad I Introducción a los lenguajes de programación y aplicaciones a ser usadas	Introducción al manejo de lenguajes de programación b. Componentes de un programa, estructura y sentencias. Introducción a Netbeans , herramienta complementaria que maneja C++	Entiende y comprende las aplicaciones necesarias para el desarrollo de lenguajes de programación	Práctica en Clase Diapositivas Videos	**5 horas**
UNIDAD II Estructura de un programa, librerías, tipo de datos, declaración de variables, impresión y lectura de datos en pantalla.	Aprenden a estructurar un programa , y comprenden la lógica para el desarrollo de un sistema	Realizan prácticas guiadas que ayudan al alumno a seguir adecuadamente la lógica matemática de un programa	Practica en clase , realizan varios ejercicios para obtener resultados en pantalla, operaciones matemáticas	**5 horas**
UNIDAD III Sentencias de Control If, Swith	La sentencia IF como Switch comprende la ejecución de instrucciones cuando la condición es verdadera, y else ejecuta las instrucciones para el caso en que la condición es falsa.	Diferencia y aplica sentencias de control para validación de ejercicios propuestos,,	Ejercicios en Clase, con ejemplos de casos reales. Debate e Prácticas en clase	**10 horas**

UNIDAD IV Sentencias de Repeticion For, While, Do-While	Introducción en el manejo de ciclos de repeticion , for, se introducen ejemplos , ejercicios, de igual manera se revisa la sentencia while y Do while	Se introducen en el beneficio del uso de los ciclos de repetición, se simplifican los códigos y estructura de programación.	Ejercicios de programación en clase, practicas guiadas en la plataforma	**5 horas**
UNIDAD V Arreglos y Estructuras	Se introducen en vectores y el almacenamiento en memoria, se coordina con estructuras y funciones para obtener un mejor desarrollo del programa.	Conoce como estructurar la información de forma organiza, toma como punto de partida para inicializar un programa con complejidad.	Ejercicios de programación en clase, practicas guiadas en la plataforma Evaluaciones Explosión de casos de estudio	**5 horas**

2. CONOCIMIENTOS PREVIOS Y RELACIONADOS

Co-requisitos

1.-Identificar los conceptos básicos de la importancia del uso de los lenguajes de programación en la educación y la aplicación en proyectos de carrera.

2.-Realizar prácticas en clase que permitan plantear ejercicios de dominio matemático e estadístico..

3.-Realizar prácticas en clase que permitan conocer lista comandos, tipos de datos verdaderos y errados. (Comprobar en la computadora)

4. Conocer las estrategias para diferenciar las variables, constantes y operadores que se aplican en el sistema informático.

5. Diseñar y ejecutar presentaciones que plantean fórmulas de ingeniería para resolver con algoritmos y codificación en C++.

3. UNIDADES TEÓRICAS

- **Desarrollo de las Unidades de Aprendizaje (contenidos)**
  - **A. Base Teórica**

UNIDAD I : Introducción a los lenguajes de programación y aplicaciones a ser usadas

TEMA 1: Introducción a los lenguajes de programación.

1. Introducción.

Es el idioma utilizado para controlar el comportamiento de una máquina, maquinaria o equipos. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones

Historia El Lenguaje C fue creado en 1972 por Dennis Ritchie en un PDP-11 de Digital Equipment Corporation bajo el sistema operativo UNIX. El lenguaje C es del tipo lenguaje estructurado como son Pascal, Fortran, Basic.

2. Instalación y configuración de C++ con Netbeans

NetBeans IDE es una herramienta muy útil permite desarrollar aplicaciones para sistemas operativos como Windows, Mac, Linux y Solaris. NetBeans es un proyecto público que consiste en un IDE (Integrated Development Enviroment) de código abierto y una plataforma de aplicación.

Pasos para la Instalacion:

- Descargar ,e instalar la versión mas reciente de NetBeens, para esta clase descargaremos Java : jdk-8u111-nb-8_2-windows-x64.exe , Cygwin: setup-x86_64.exe.
- Mientras se realiza la instalación de Cygwin se debe considerar seleccionar los paquetes:
    - gcc-core
    - gcc g++
    - Devel:
    - make GNU
    - Devel:
    - gdb GNU
    - Cmake
    - qmake

- Editar Variables de entorno - en propiedades del sistema Buscar en Variables del sistema :
  Editar la variable del sistema. No borrar añadir un ; al final y agregar la dirección del cygwin:
  Path : \;C:\cygwin64\bin\;C:\cygwin64\lib\;

- Configurar programa de Netbeans

  Ingresar al Menu Tools -
  Options C++
  Add New TOOL Collection
  Buscar ruta:
  C:\cygwin64\bin

Seleccionar la collection: GNU 4.x CYgwin

C: compiler :

C:\cygwin64\bin\gcc.exe C++

compiler : C:\cygwin64\bin\g++.exe

Assembler : C:\cygwin64\bin\as.exe

Make Command:

C:\cygwin64\bin\make.exe Debugger

Command: C:\cygwin64\bin\gdb.exe

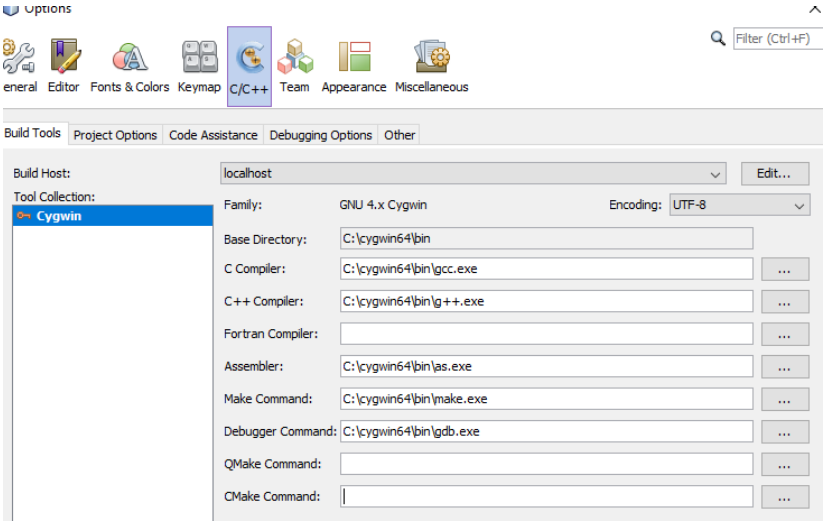


Figura1: Configuración de c++ en Netbeans

3. Creación de Nuevos Proyectos C++

   Para la creacion de nuevos proyectos se debe Ingresar al menú File/ New Project C++ Aplication.

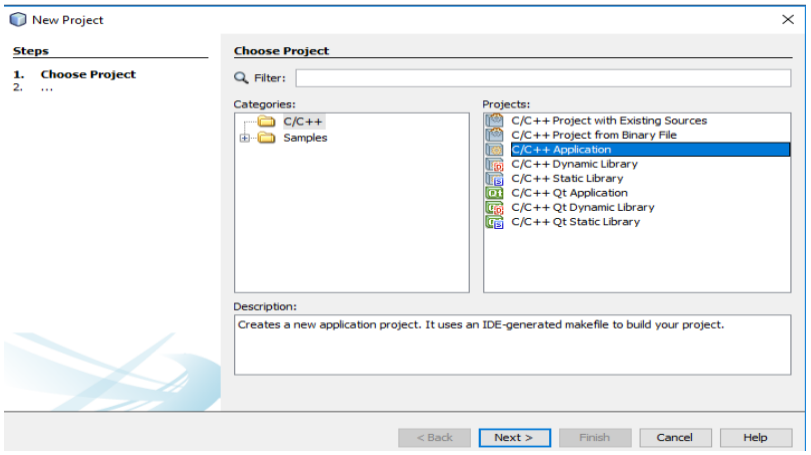


Figura 2: Imagen de cómo crear nuevos proyectos

- Dar un nombre al proyecto y seleccionar la ubicación donde se alojarán los archivos

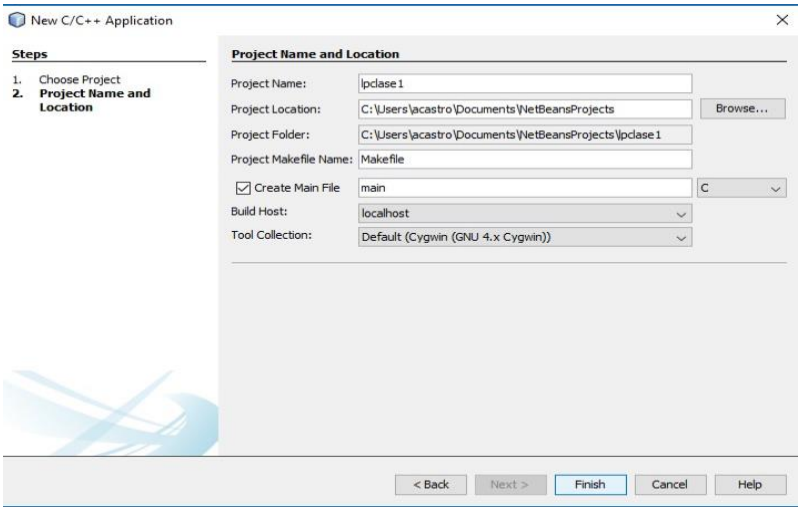


Figura 2: Guardar Proyecto

- Configurarla salida en pantalla del programa Dar click derecho sobre el proyecto y seleccionar propiedades, Buscar la opcion Run y cambiar a consola: Standard output

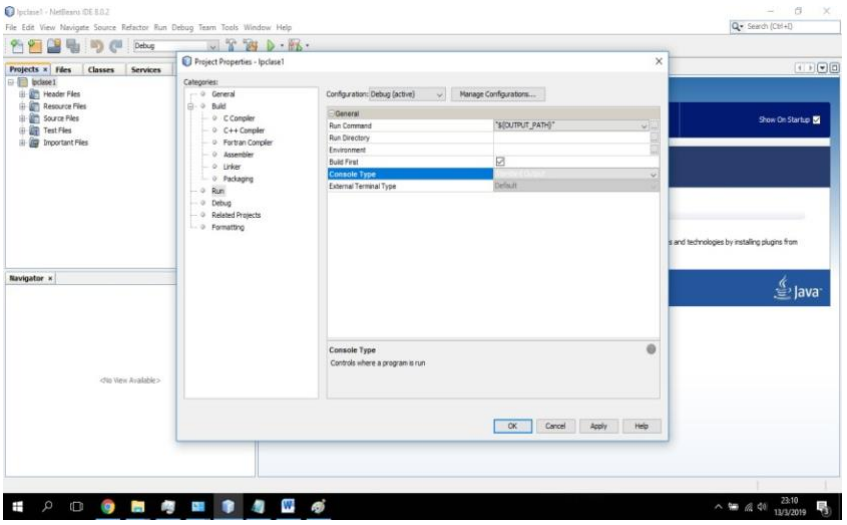


Figura 3: Configuración la salida de pantalla

- Compilación del programa

  Iniciar creando un programa que imprima “Hola mundo” , a continuación se detallan las opciones para compilar un programa

  - Menú Run
  - Compilar F9
  - Construir F11
  - Ejecutar F6

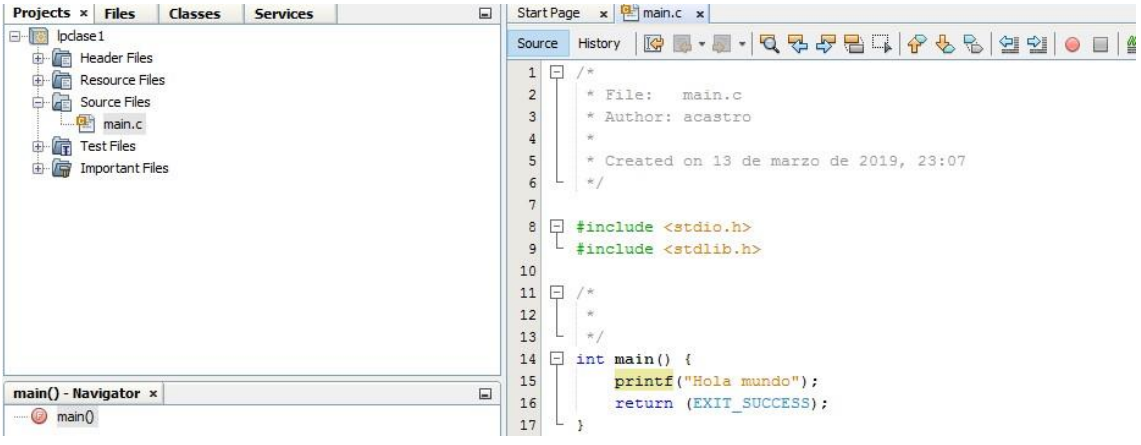


Figura 4: Configuración la salida de pantalla Netbeans

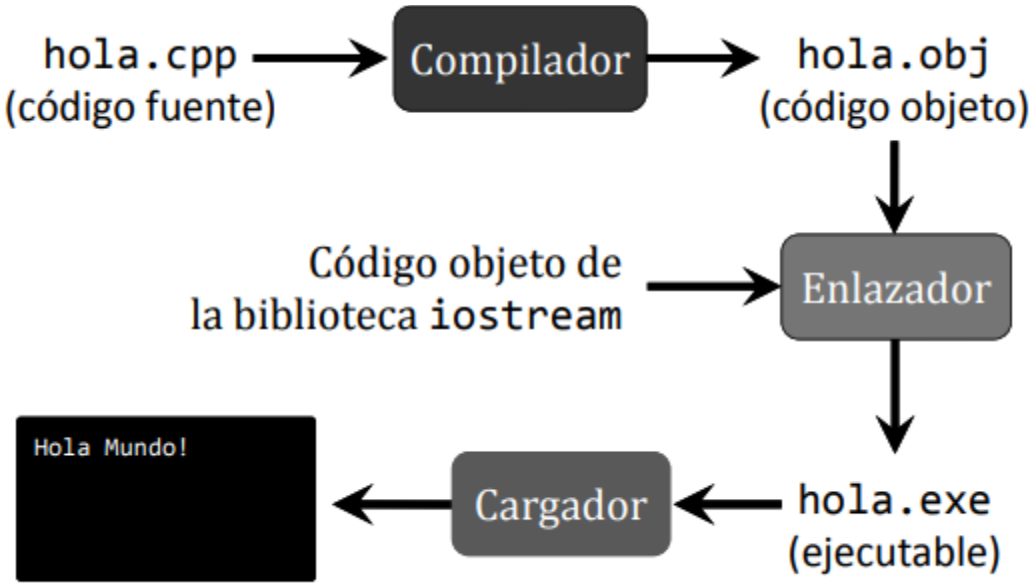


Figura 5: Salida de archivos cpp

UNIDAD II : Estructura de un programa, librerías, tipo de datos, declaración de variables, impresión y lectura de datos en pantalla.

TEMA 1: Sintaxis de un Lenguaje de Programación

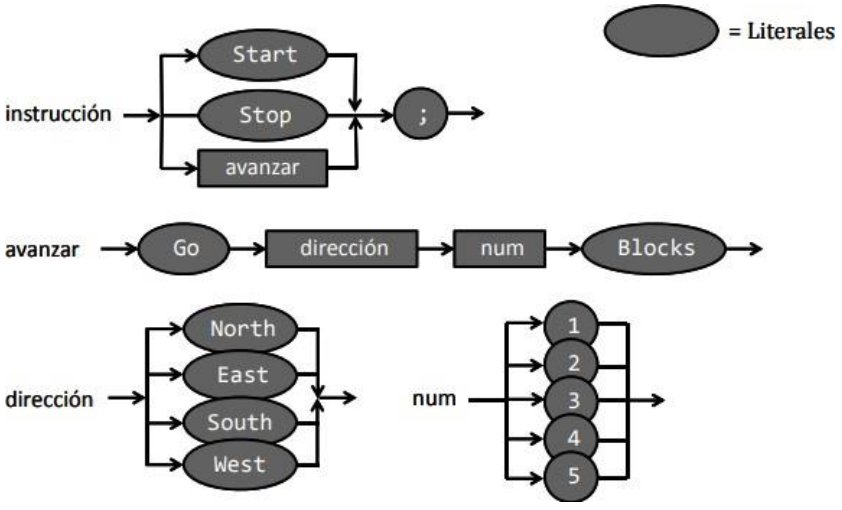


Figura 6: Sintaxis de un lenguaje de programación

TEMA 2: Tipo de Datos

- Los tipos de datos en C++ se clasifican en primitivos y derivados.
- Los tipos de datos primitivos son los que están definidos dentro del lenguaje.
- Los tipos de datos derivados se forman a partir de los tipos primitivos.

En este tema veremos los tipos primitivos y en temas siguientes estudiaremos los tipos derivados. Los tipos de datos primitivos en C++ son: numéricos enteros, numéricos reales, tipo lógico y tipo carácter ampliado.

Tipos de datos C++ numéricos enteros

El tipo de dato numérico entero es un subconjunto finito de los números enteros del mundo real. Pueden ser positivos o negativos.

En C++ los tipos de datos numéricos enteros son los siguientes:

Tipo de Dato	Descripción	Número de bytes típico	Rango
short	Entero corto	2	-32768 a 32767
int	Entero	4	-2147483648 a +2147483647
long	Entero largo	4	-2147483648 a +2147483647
char	Carácter	1	-128 a 127

Con los tipos enteros pueden utilizarse los calificadores signed y unsigned. Estos calificadores indican si el número tiene signo o no. Si se usan solos, sin indicar el tipo de dato se asume int.

Por ejemplo, las siguientes declaraciones son equivalentes: unsigned int x; equivale a: unsigned x;

Usando estos calificadores podemos tener los siguientes tipos enteros:

Tipo de Dato	Descripción	Número de bytes típico	Rango
signed short	Entero corto	2	-32768 a 32767
unsigned short	Entero corto sin signo	2	0 a 65535
signed int	Entero	4	-2147483648 a +2147483647
unsigned int	Entero sin signo	4	0 a 4294967295
signed long	Entero largo	4	-2147483648 a +2147483647
unsigned long	Entero largo sin signo	4	0 a 4294967295
signed char	Carácter	1	-128 a 127
unsigned char	Carácter sin signo	1	0 a 255

Podemos ver que los datos enteros de tipo signed son equivalentes a los enteros sin utilizar el calificador:
signed int a; es equivalente a escribir int a;

Tipos de datos numéricos reales

El tipo de dato numérico real es un subconjunto finito de los números reales. Pueden ser positivos o negativos.
En C++ los tipos de datos numéricos reales son los siguientes:

Tipo de Dato	Descripción	Número de bytes típico	Rango
float	Real (Número en coma flotante)	4	Positivos: 3.4E-38 a 3.4E38 Negativos: -3.4E-38 a -3.4E38
double	Real doble(Número en coma flotante de doble precisión)	8	Positivos: 1.7E-308 a 1.7E308 Negativos: -1.7E-308 a -1.7E308
long double	Real doble largo	10	Positivos: 3.4E-4932 a 1.1E4932 Negativos: -3.4E-4932 a -1.1E4932

Tipo lógico

Los datos de este tipo sólo pueden contener dos valores: true ó false (verdadero ó falso). Si se muestran como enteros, el valor true toma el valor 1 y false el valor 0.

Tipo de Dato	Descripción	Número de bytes típico	Rango
bool	Dato de tipo lógico	1	0, 1

Tipo carácter extendido

Este tipo se utiliza para representar caracteres UNICODE. Utiliza 2 bytes a diferencia del tipo char que solo utiliza 1.

Tipo de Dato	Descripción	Número de bytes típico	Rango
wchar_t	Carácter Unicode	2	0 a 65535

TEMA 3: **Declaración de variables en un programa C y C++**

Para tener una variable en un programa, previamente hay que definirla con un tipo y un identificador (es decir, un nombre para la variable).

Por ejemplo:
int numero define una variable de tipo entero que se llama numero. Y seguidamente hay que añadir un punto y coma a la expresión:
int numero;
El punto y coma transforma la expresión en una instrucción que se ejecutará en el programa. Cuando se llegue a ella, el programa reservará espacio de memoria según el tipo de variable para poder escribir y leer valores.
Por lo tanto, la fórmula general es:
<tipo> <identificador> <;>
Es lo que llamamos una declaración de variable. En C y C++ todas las variables se deben declarar antes de su uso. Si no, se producirá un error de compilación.
A continuación se muestra un programa completo que funciona: solo declara variables de tipos distintos:

```
int main()
{
char
caracter;
short valor;
int numero;
long numeroMasLargo;
float numeroRealFloat;
double
numeroRealDoble;

// fin
return 0;
}
```

Existe un método para acceder a una variable global enmascarada por una variable local. Se trata del operador de ámbito, que consiste en dos caracteres de dos puntos seguidos (::).

Veremos este operador con más detalle en el capítulo dedicado a los espacios con nombre, pero veamos ahora cómo lo podemos usar para acceder a una variable global enmascarada:

```
int x; // Variable global

int main()
{
   int x; // Variable local que enmascara a la global

   x = 10; // Accedemos a la variable local
   ::x = 100; // Mediante el operador de ámbito accedemos a la global
   return 0;
}
```

Unidad III : Sentencias de Control C++

TEMA 1: **Condicionales**

Hay tres tipos de sentencia condicional: if-else-if, ? y switch

1. if-else-if

f en inglés significa si (condicional, no sí de afirmación) y la estructura es la siguiente:

```
1  if (condición)
2  {
3
4          lo que se ejecuta si se cumple la condición
5  }
```

Entre paréntesis va la condición que se expresa con operadores y comparadores que vimos en la entrega anterior. Por ejemplo, vamos a hacer un programa que crea dos variables de tipo “int”, deja que el usuario meta un número en cada una y luego dice cuál es mayor.

Ejemplo:

```
int main()
{
    int edad;

    cout << "Escribe cuántos años tienes: ";
    cin >> edad;

    //Indica si el usuario es mayor de edad

    if (edad >= 18) {
    cout << "Puedes pasar";
    }

    //Indica si el usuario es mayor de edad
    if (edad < 18) {
    cout << "Acceso denegado, eres menor de edad";
    }
```

Figura 7 : Ejemplo con la sentencia IF

Un else después de un if significa: si la condición del if no se ha cumplido, entonces ejecuta esto:

```
if (condición)
{
        lo que se ejecuta si se cumple la condición
}
else
{
      lo que se ejecuta si no se cumple la condición
}
```

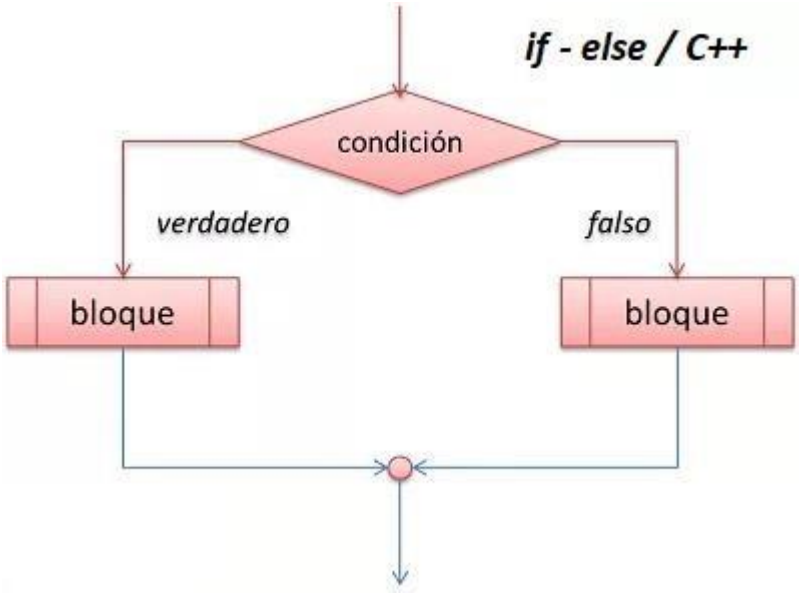


Figura 8 : Estructura IF-ELSE

La sentencia if / else controla las 2 posibilidades de una decisión, if ejecuta las instrucciones cuando la condición es verdadera, y else ejecuta las instrucciones para el caso en que la condición es falsa.

La sintaxis de la estructura if / else es la siguiente:

```
if (condición) {
 //Instrucciones cuando la condición es verdadera
 } else {
 //Instrucciones cuando la condición es falsa
}
```

En donde,
(condición), es una expresión que da como resultado un valor lógico (verdadero, falso), los paréntesis agrupando a la condición son obligatorios.

{ }, las llaves indican el alcance de la expresión de if y de la expresión else, respectivamente, son necesarias a menos que la instrucción esté formada solo por una sentencia.

//sentencias, las instrucciones que corresponden a la primera parte se ejecutarán si la condición es verdadera. Si la condición es falsa se ejecutarán las instrucciones que se encuentran enseguida del else.

Reescribiendo el programa anterior utilizando if / else queda de la siguiente forma:

```
#include <iostream>

using namespace std;

int main()
{
    int edad;

    cout << "Escribe cuántos años tienes: ";
    cin >> edad;

    //Indica si el usuario es mayor de edad

    if (edad >= 18) {
        cout << "Puedes pasar";
    }
    else {
        cout << "Acceso denegado, eres menor de edad";
    }

    return 0;

}
```

Figura 9 : Ejemplo de IF-ELSE

2.- **SWITCH**

Este último tipo se utiliza cuando se quiere hacer algo para cada valor de una variable pero hay demasiados valores por lo que hacer un if else if sería muy incómodo.

El uso de switch es el siguiente:

```
switch (variable)
{
        case valor1:
               Lo que se ejecuta si la variable tiene el valor1
               break; //sale del switch
        case valor2:
               Lo que se ejecuta si la variable tiene el valor2
               break;
        case valorN;
               Lo que se ejecuta si la variable tiene el valorN
               break;
        default:
               Lo que ejecuta si la variable no tiene ninguno de l
               break;
}
```

Ejemplo :

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Ingrese la Opción a ejecutar: ";
    int opcion = 0;
    cin >> opcion;

    switch(opcion)
    {
        case 1: cout << "Usted ha seleccionado la opción 1";
        break;
        case 2: cout << "Usted ha seleccionado la opción 2";
        break;
        case 3: cout << "Usted ha seleccionado la opción 3";
        break;
        default: cout << "Usted ha ingresado una opción incorrecta";
    }
    system("PAUSE");
}
```

Figura 10 : Ejemplo de SWITCH

Unidad IV : Sentencias de Repetición

TEMA 1: Sentencia FOR

El bucle for nos permite repetir una tarea un número de veces, el formato es el siguiente:

```
for ( dar valores ; condicion ; lo que hacer )
{
lo que se ejecuta
}
```

Por ejemplo, vamos a hacer un programa que imprima en la pantalla 10 veces la frase Hola Mundo con el número de frase antes. Esto ante lo teníamos que hacer escribiendo 10 veces una función printf con el mensaje, pero ahora podemos usar el bucle for para hacerlo de manera más sencilla:

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i; //Usaremos esta variable para contar el número de veces

    for (i = 0; i < 10; i++)
    {
        printf("%i.- Hola mundo\n", i+1);
    }

    return 0;
}
```

Es un código bastante sencillo, lo primero es dar valores. Lo que pongamos aquí solo se ejecutará al comenzar el bucle, y le decimos que i sea igual a 0. Lo segundo es la condición y lo tercero es lo que hacer si la condición se cumple. Si la condición se cumple entonces se ejecuta lo tercero y se da paso al bloque de código que hay entre { y }. Si no se cumple entonces se termina el bucle y se sigue con el código. Si i es menor de 10 le suma

1 a i y hace esa función printf, en cuanto i sea mayor o igual a 10, ya no se ejecutará el bucle, como cada vez le sumamos 1 (i++) se ejecutará diez veces.

Ejercicios Sentencia For:

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

int main(void){
    /*Declaro las variables*/
    int i=0, com=0, n=0;total=0
    system("cls");
    cout<<"Escriba el numero total de empleados: ";
    cin>> n;
    cout<<"Escriba el vamor de comision ";
    cin>> com;

    for (i=1;i<=n;++i){
        total=com/n;

    }

    /*Muestra el resultado en pantalla*/
    cout<<"El valor a recibir por empleado es: "<<total;
    system("pause");
}
```

Figura 11 : Ejemplo sentencia FOR

TEMA 2: **Sentencia WHILE**

La sentencia while es parecido a for, solo que en este simplemente especificamos la condición, es útil si queremos hacer un bucle indefinido, su uso es:

```
1  while(condición)
2  {
3      a ejecutar;
4  }
```

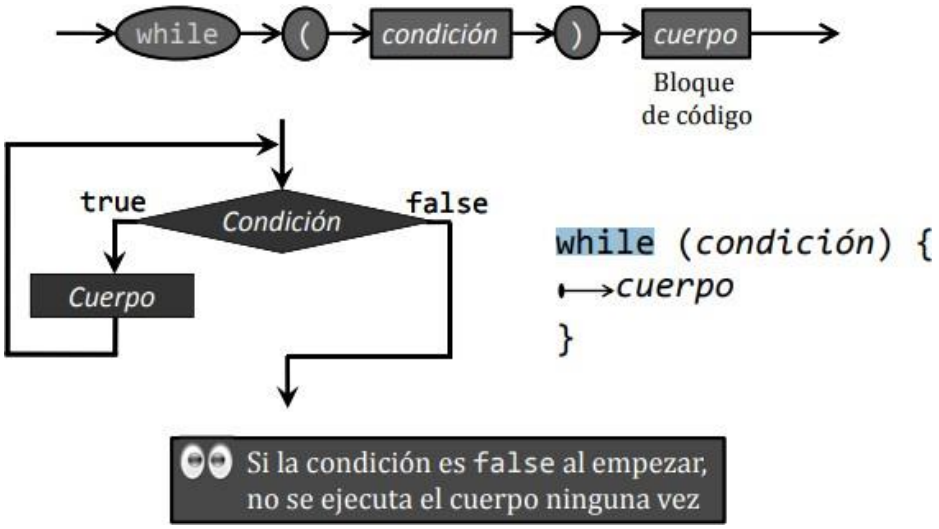


Figura 12 : Manejo sentencia while

Ejemplo:

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

int main() {
   int i = 1, n = 0, suma = 0;
   while (n <= 0) { // Sólo n positivo
      cout << "¿Cuántos números quieres sumar? ";
      cin >> n;
   }
   while (i <= n) {
      suma = suma + i;
      i++;
   }
   cout << "Sumatorio de i (1 a " << n << ") = "
        << suma << endl;
   return 0;
}
```

$$\sum_{i=1}^{n} i$$

```
D:\FP\Tema02>serie
¿Cuántos números quieres sumar? -3
¿Cuántos números quieres sumar? 0
¿Cuántos números quieres sumar? 5
Sumatorio de i (1 a 5) = 15
```

Figura 13 : Ejemplo sentencia while

Ejercicio:

```
#include <iostream>

using namespace std;
int main()
{
    int numero;
    cout <<  "Ingrese un numero ";
    cin >> numero;
    while(numero <= 100)
    {
        cout <<  "Ingrese un numero ";
        cin >> numero;
    }
    system("PAUSE");
    return 0;
}
```

Figura 14 : Ejercicio sentencia while

TEMA 3: Sentencia DO-WHILE

Es similar a while, solo que este primero ejecuta y luego comprueba la condición.

```
1  do
2  {
3      a ejecutar;
4  } while (condición);
```

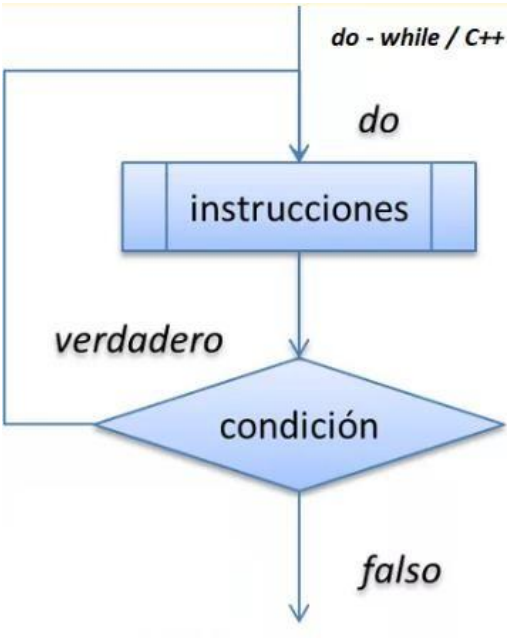


Figura 15 : sentencia DO-while

do, palabra reservada que indica el inicio del ciclo

{}, Todas las instrucciones dentro de las llaves se ejecutarán como una instrucción como parte del ciclo while, como parte del código debemos generar la instrucción de paro de modo que podamos salir de forma limpia del ciclo.

while, palabra reservada que establece el ciclo.

(condición de parada), es una condición lógica que se evalúa por la instrucción while, en caso de ser verdadera, se ejecutarán las instrucciones dentro del ciclo while, en caso de ser falsa, se ignora este código.

El siguiente ejemplo ilustra el uso del ciclo do – while

El bucle do – while se puede leer como «haz – mientras», esta estructura se encarga de repetir de forma cíclica un conjunto de instrucciones que se encuentren dentro del bucle, esta repetición se realizará hasta que se cumpla la condición de parada que definamos y que es evaluada con la palabra reservada while

Ejercicio do –while

```
#include <iostream>

using namespace std;

int main()
{
    int a = 0;

    cout<<"Tabla de códigos ASCII  utilizando do While" << endl  << endl;

    cout << "Código \t Caracter" << endl;
    do {

        // char(a) convierte el valor numérico de "a" a su correspondencia ASCII
        cout << a << " \t - " << char(a) << endl;

        a = a + 1;  //incrementamos nuestro contador


    } while (a<=255);

    return 0;
}
```

Figura 16 : Ejercicio sentencia DO-while

Unidad IV : Arreglos y Estructuras

TEMA 1: Arreglos

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.

Los arrays permiten agrupar datos usando un mismo identificador. Todos los elementos de un array son del mismo tipo, y para acceder a cada elemento se usan subíndices.

Sintaxis:
<tipo> <variable de array>[<número de elementos>][[<número de elementos>]...];

Los valores para el número de elementos deben ser constantes, y se pueden usar tantas dimensiones como queramos, limitado sólo por la memoria disponible.
Cuando sólo se usa una dimensión se suele hablar de listas o vectores, cuando se usan dos, de tablas.

Ahora podemos ver que las cadenas de caracteres son un tipo especial de arrays. Se trata en realidad de arrays de una dimensión de tipo char.
Los subíndices son enteros, y pueden tomar valores desde 0 hasta <número de elementos> -1. Esto es muy importante, y hay que tener mucho cuidado, por ejemplo:

```
int Vector[10];
```

Creará un array con 10 enteros a los que accederemos como Vector[0] a Vector[9].
Como subíndice podremos usar cualquier expresión entera.

En general C++ no verifica el ámbito de los subíndices. Si declaramos un array de 10

elementos, no obtendremos errores al acceder al elemento 11. Sin embargo, si asignamos valores a elementos fuera del ámbito declarado, estaremos accediendo a zonas de memoria que pueden pertenecer a otras variables o incluso al código ejecutable de nuestro programa, con consecuencias generalmente desastrosas.

Ejemplo:

```
int Tabla[10][10];
char DimensionN[4][15][6][8][11];
...
DimensionN[3][11][0][4][6] = DimensionN[0][12][5][3][1];
Tabla[0][0] += Tabla[9][9];
```

Cada elemento de Tabla, desde Tabla[0][0] hasta Tabla[9][9] es un entero. Del mismo modo, cada elemento de DimensionN es un carácter.

Asignación de arrays:

Los arrays pueden ser inicializados en la declaración.

Ejemplos:

```
float R[10] = {2, 32, 4.6, 2, 1, 0.5, 3, 8, 0, 12};
float S[] = {2, 32, 4.6, 2, 1, 0.5, 3, 8, 0, 12};
int N[] = {1, 2, 3, 6};
int M[][3] = { 213, 32, 32, 32, 43, 32, 3, 43, 21};
char Mensaje[] = "Error de lectura";
```

En estos casos no es obligatorio especificar el tamaño para la primera dimensión, como ocurre en los ejemplos de las líneas 2, 3, 4 y 5. En estos casos la dimensión que queda indefinida se calcula a partir del número de elementos en la lista de valores iniciales.

En el caso 2, el número de elementos es 10, ya que hay diez valores en la lista.

En el caso 3, será 4.

En el caso 4, será 3, ya que hay 9 valores, y la segunda dimensión es 3: 9/3=3.

Y en el caso 5, el número de elementos es 17, 16 caracteres más el cero de fin de cadena.

TEMA 2: Estructuras anidadas

También está permitido anidar estructuras, con lo cual se pueden conseguir superestructuras muy elaboradas.

Ejemplo:

```
struct stDireccion {
char Calle[64];
int Portal;
int Piso;
char Puerta[3];
char CodigoPostal[6];
char Poblacion[32];
};
struct stPersona {
struct stNombre {
char Nombre[32];
char Apellidos[64];
} NombreCompleto;
stDireccion Direccion;
char Telefono[10];
      };
```

En general, no es una práctica corriente definir estructuras dentro de estructuras, ya que resultan tener un ámbito local, y para acceder a ellas se necesita hacer referencia a la estructura más externa.

Por ejemplo para declarar una variable del tipo stNombre hay que utilizar el operador de acceso (::):

stPersona::stNombre NombreAuxiliar;

Sin embargo para declarar una variable de tipo stDireccion basta con declararla:

stDireccion DireccionAuxiliar;

EJERCICIOS

//Desarrolle una estructura que guarde los datos de "N" alumnos de la materia de Estructuras de Datos, y los muestres por pantalla, la estructura debe tener
//(cedula, nombre, apellido, edad, profesión, lugar de nacimiento, dirección y teléfono).

```
#include <iostream>
#include <windows.h>
#define max 100 /*Constante*/
using namespace std;

int main(void){
    /*Declara las variables para los ciclo for*/
    int i = 0, n = 0;

    /*Declara estructura persona*/
    struct persona{
        char cedula[14];
        char nombre[15];
        char apellido[15];
        int edad;
        char profesion[20];
        char lugar[50];
        char direccion[50];
        int telefono;
    };
    /*Declara alumno, arreglo de la estructura persona*/
    struct persona alumno[max];

    /*Se pide cuantos registros de alumnos se guardaran*/
    cout<<"Cuantos datos desea introducir?/n";
    cin>> n;

    /*Ciclo for que va a recorrer según la cantidad escrita anteriormente*/
```

```
for (i = 0; i < n; i++){
    cout<<"\nEscriba la Cedula "<< i+1<<":";
    cin>> alumno[i].cedula;

    cout<<"\nEscriba el Nombre "<< i+1<<":";
    cin>> alumno[i].nombre;

    cout<<"\nEscriba el Apellido "<< i+1<<":";
    cin>> alumno[i].apellido;

    cout<<"\nEscriba la Edad de "<< i+1<<":";
    cin>> alumno[i].edad;

    cout<<"\nEscriba la Profesion de "<< i+1<<":";
    cin>> alumno[i].profesion;

    cout<<"\nEscriba el Lugar de Nacimiento de "<< i+1<<":";
    cin>> alumno[i].lugar;

    cout<<"\nEscriba la Direccion de "<< i+1<<":";
    cin>> alumno[i].direccion;

    cout<<"\nEscriba el Telefono de "<< i+1<<":";
    cin>> alumno[i].telefono;
}

cout<<"/nEl registro de Alumnos que se introdujeron son: \n\n";

/*Ciclo for que muestra el listado de registro ingresados*/
for (i = 0; i < n; i++){
    /*Se llama al arreglo alumno seguido de la variable cedula*/
    cout<< alumno[i].cedula;
    cout<<"\t"<<alumno[i].nombre;
    cout<<"\t"<<alumno[i].apellido;
```

```
        cout<<"\t"<<alumno[i].edad;
        cout<<"\t"<<alumno[i].profesion;
        cout<<"\t"<<alumno[i].lugar;
        cout<<"\t"<<alumno[i].direccion;
        cout<<"\t"<<alumno[i].telefono<<"\n\n";
    }

    system("pause");
}
```

Ejercicios de programación:

1) Ejercicio 1: Ejecutemos el siguiente código e interpretemos la solución, cual fue la finalidad del ejercicio. Elimine un ciclo for , ejecute y mire que ha pasado. Capture su solución

```
#include<iostream>

using namespace std;

int i,j,size;

int main() {
    cout<<"Dame el tamaño"<<endl;
    cin>> size;
    for (i=1;i<=size;i++) {
        for (j=1;j<=i;j++)
            cout << "*";
        cout<<endl;
    }
}
```

2) Ejecute el programa , capture y pegue la salida e interprete que hace el programa

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

int main ()
{
        int x, bono = 0, sueldo = 0;
 cout<<"Ingrese las horas trabajadas: ";
 cin>> x;


 if (x < 40)
 {
  sueldo = x * 20;
  cout<<"Su sueldo esta semana es de : "<<sueldo;
 }
 else
 {
  bono = x - 40;
  sueldo = (x - bono) * 20;
  cout<<"Su sueldo esta semana es de : "<<sueldo + (bono * 25);
 }

 return 0;
}
```

3) Ejecute el programa , capture y pegue la salida e interprete que hace el programa

Hacer un **programa en C++** para una tienda, donde se considera un descuento por compra a sus clientes con membresía dependiendo de su tipo, sólo existen tres tipos de membresía, tipo A, tipo B y tipo C. Los descuentos son los siguientes:

Tipo A 10% de descuento
Tipo B 15% de descuento
Tipo C 20% de descuento

Nota: El programa tiene errores en su código, rectifíquelos y explique para que sirve y porque se usa el tipo de dato float

```
#include <iostream>
using namespace std
int main ()
{
 float x;  char tipo;
 cout<<"Ingrese el monto de compra: ";
 cin>> x;
 cout<<"Tipo de cliente: ";
 cin> tipo;
 if (tipo == 'A')
 {
  cout<<"Tipo A" endl;
  cout<<"Total a pagar: "<<x - (x * 0.10)
 }
 else if (tipo == 'B')
 {
  cout<<"Tipo B"<<endl;
  cout<<"Total a pagar: "<<x - (x * 0.15);
 }
 else if (tipo == 'C')
 {
  cout<<"Tipo C"<<endl
  cout<<"Total a pagar: "<<x - (x * 0.20);
 }
 else
 {
  cout<<"Total a pagar: "<<x;
 }
 return 0;
}
```

Realizar un programa que ingrese n numero de clientes si el valor de compra que realiza es en efectivo se le otorga un 10% de descuento, caso contrario un incremento del 4%

```
#include <iostream>
#include <string>

using namespace std;

int main()

{
   int valorcompra,nclientes,total;
   string tipopago;
   cout<<" Ingrese el numero de clientes que realiza su compra"<<endl;
   cin>>nclientes;
```

```
for (int i=1;i<=nclientes;i++)
   {
      cout<<"Ingrese el valor de la compra del cliente "<<i<<endl;
      cin>>valorcompra;
      cout<<"Ingrese el tipo de de pago que reliza el cliente "<<i<<endl;
      cin>>tipopago;
      if (tipopago=="efectivo")
      {
         total=valorcompra-(valorcompra*0.10);
         cout<<"El total de la compra de cliente "<<i<<"con valor de descuento es de : "<<total<<endl;
      }
       else
      {
          total=valorcompra+(valorcompra*0.04);
         cout<<"El total de la compra de cliente "<<i<<"con valor de incremento es de : "<<total<<endl;
      }

   }
}
```

1) Ejecute el programa , interprete los resultados y capture imágenes

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

int main ()
{
 float salario, aumento;
 char contrato;
cout<<"indique la vocal que represente su tipo de contrato"<<endl;
cout<<"a) De 0 a $399"<<endl;
cout<<"b) De $400 a $600"<<endl;
cout<<"c) $601 a $800"<<endl;
cout<<"d) Mas de $800"<<endl;
cin>>contrato;
cout<<"Ingrese su salario actual"<<endl;
cin>>salario;

 switch (contrato)
 {
  case 'a': case 'A':
  aumento = salario + (salario * 0.2);
  cout<<"Su nuevo salario sera: "<<aumento<<endl;
  break;
  case 'b': case 'B':
  aumento = salario + (salario * 0.1);
  cout<<"Su nuevo salario sera: "<<aumento<<endl;
  break;
  case 'c': case 'C':
  aumento = salario + (salario * 0.05);
  cout<<"Su nuevo salario sera: "<<aumento<<endl;
  break;
  case 'd': case 'D':
  aumento = salario + (salario * 0.03);
  cout<<"Su nuevo salario sera: "<<aumento<<endl;
  break;
  default:
  cout<<"Opcion no valida";
 }

return 0;
}
```

2) Ejecute y transforme el programa de la imagen con la sentencia swicth, es decir en la opción
   1: Suma
   2: Resta
   3: Multiplicación

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
    float n1, n2;
    float suma, resta, mult, divi;
    cout << "\nIntroduzca un numero: ";
    cin >> n1 ;
    cout << "\nIntroduzca otro numero: ";
    cin >> n2;
    suma = n1 + n2;
    resta = n1 - n2;
    mult = n1 * n2;
    cout << fixed << setprecision(2);
    cout << "\nsuma: " << fixed << setprecision(2) << suma;
    cout << "\nresta: " << resta;
    cout << "\nmultiplicacion: " << mult;
    system("pause");
}
```

3) Ejecute , repase e investigue para que sirve la sentencia DO-WHILE

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

int main() {
 int uni, precio;
 do{
 cout << "Introduce el número de unidades a comprar: ";
 cin >> uni;
 }while(uni < 1);

 switch(uni){
 case 1: precio = 100;
 break;
 case 2: precio = 2*95;
 break;
 case 3: precio = 3*90;
 break;
 default: precio = uni * 85;
 break;
 }

 cout << "El precio de la compra es "<< precio;
}
```

B. Base de Consulta

TÍTULO	AUTOR	EDICIÓN	AÑO	IDIOMA	EDITORIAL
APRENDA C++	Javier Garcia de Jalon.	1	1998	Español	Universidad de Navarra
El C++ por la pràctica Introducción al lenguaje y su filosofía	Lluís Gil Espert Montserrat Sánchez Romero	1	1999	Español	Ediciones de la Universitaria Politécnica de Catalunya
La ruta práctica a Power Point 2007	Editora Macro E.I.R.L	1	2007	Español	Macro E.I.R.L

Programación en C/C++	**Fermí Vilà**	1	2011	Español	Buenos Aires Paidós
Orientación educativa y TIC	Domínguez Fernández, Guillermo	1	2012	Español	Bogotá Ediciones de la U 2012
Manual imprescindible Microsoft Word 2013	Charte Ojeda, Francisco	1	2013	Español	Madrid Anaya Multimedia

C. Base práctica con ilustraciones

TRABAJOS A PRESENTAR	DETALLE
	3- Presentación del Proyecto Impreso Debe contener carátula, encabezado y pie de página , Índice, inicio a normas APA

4. ESTRATEGIAS DE APRENDIZAJE

ESTRATEGIA DE APRENDIZAJE 1: Análisis y Planeación
Descripción: La complejidad de los programas que se desarrollan actualmente produce la necesidad de iniciar a los alumnos en un camino que los conduzca a utilizar efectivas técnicas de programación. Es importante para ello poner énfasis en el diseño previo. Como se ha comprobado, una estrategia valedera es comenzar a enseñar programación utilizando los algoritmos como recursos esquemáticos para plasmar el modelo de la resolución de un problema. Esto genera una primera etapa de la programación que resulta un tanto tediosa para los alumnos que están ávidos de utilizar

la computadora. Si bien no aparecen dificultades graves con el aprendizaje de esta técnica, se puede comprobar que no resulta una tarea trivial obtener un algoritmo semánticamente correcto. El hecho de reescribir los algoritmos hasta ponerlos a punto es operativamente complicado cuando se trabaja con lápiz y papel. Además, comprobar la corrección del algoritmo presenta inconvenientes importantes. Es difícil, mental o gráficamente, representar las acciones del algoritmo en ejecución de manera totalmente objetiva, sin dejarse llevar por la subjetividad, fundamentalmente cuando el que lo hace es el propio autor del algoritmo. Por otra parte, se ha comprobado que el uso del método global para el aprendizaje del lenguaje de programación, ahorra tiempo y esfuerzo. Con el propósito de trabajar especialmente sobre los aspectos mencionados se creó un Ambiente de Aprendizaje con un editor interactivo de algoritmos, un constructor automático de trazas y un traductor de algoritmos a programas en lenguaje Pascal. Se presentan en este trabajo, los resultados obtenidos en una experiencia de campo diseñada para comprobar la efectividad de la aplicación del entorno de programación mencionado.

Se pretende adquirir conocimiento , a través de la práctica participativa en temas relevantes como : Introducción a los Algoritmos, Conceptos básicos, Identificadores, entradas y salidas
Estructuras de Control Selectivas, Primera Práctica, Estructura de Control Repetitiva While y Do While. Estructura de Control Repetitiva For, Segunda Práctica, Funciones, Vectores y Funciones con y sin parámetros. Matrices, Tercera Práctica, Ordenamiento y Examen Final. Los ejercicios en clase guiados por el maestro ayudan al estudiante a comprender el lenguaje de mejor manera y genera a su vez la necesidad de poder crear nuevo conocimiento, investigar nuevos conceptos complementarios al buen desempeño del trabajo que se está realizando.
Se persigue que el estudiante debe mostrar interés en el desarrollo del curso, asumir un compromiso en las pautas metodológicas, participar activamente en modo personal y grupo.
Finalmente el estudiante se identifica con la carrera al plantear sus propias aplicaciones para su solución. Participa activamente con responsabilidad y respeto.

Ambiente(s) requerido:

Aula amplia con buena iluminación.

Material (es) requerido:

Infocus.

Laboratorios de computación

Parlante
Docente: Con conocimiento de la materia.

5. ACTIVIDADES

- Controles de lectura
- Exposiciones
- Presentación del Trabajo final
- CD con contenido del Proyecto
- Habilidad y esfuerzo en el proyecto entregado

6. EVIDENCIAS Y EVALUACIÓN

Tipo de Evidencia	**Descripción (de la evidencia)**
De conocimiento:	Creación de un sistema informático aplicando una metodología de enseñanza proporcionada basada en normas APA , que permitirá al estudiante comprender como comercializar y promocionar un producto apoyado de las TIC's.
Desempeño:	Trabajo individual presentación del trabajo sobre la creación de un sistema de control para usuarios finales. Exposición individual del proyecto educacional
De Producto:	✓ Trabajo de realizado consiste en la presentación de un programa completo desarrollado en C++. ✓ Defensa del proyecto mediante una práctica de los estudiantes Exposiciones orales sobre los temas de investigación individuales asignados a los estudiantes, sobre hojas de cálculo
De Innovación	Programas creados , para solución tecnológica empresarial

Criterios de Evaluación (Mínimo 5 Actividades por asignatura)	Identificar la lógica matemática aplicada en un sistema informático.

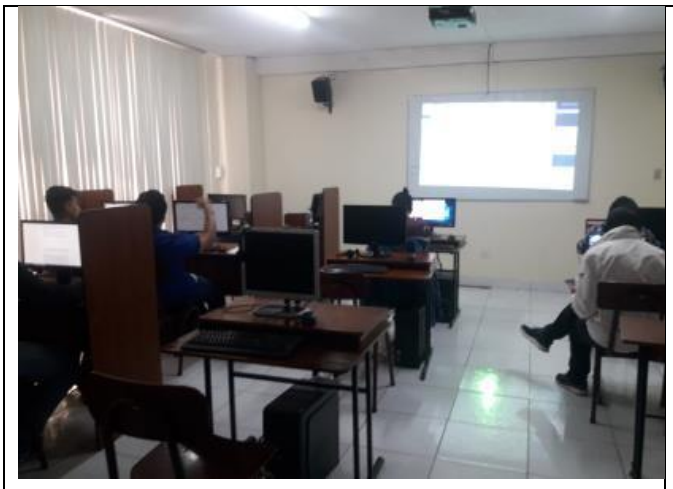
+

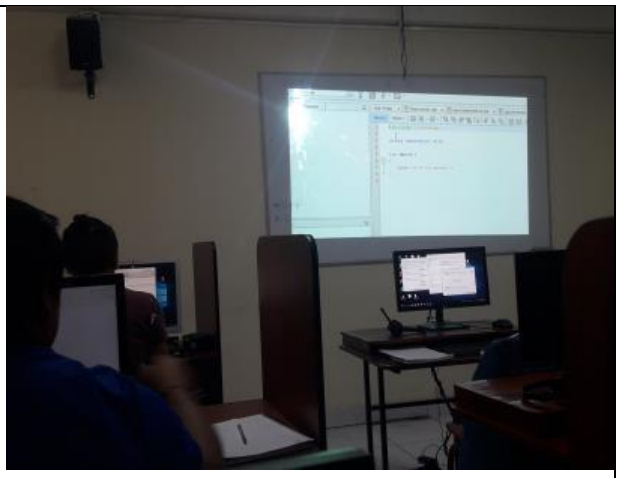
ANEXO 1 EVIDENCIAS DE APRENDIZAJE

Msc. Diana Moncayo	**Alexis Benavides**	**Milton Altamirano**
Elaborado por: **(Docente)**	**Revisado Por:** **(Coordinador)**	**Reportado Por:** **(Vicerrector)**

ANEXO 1

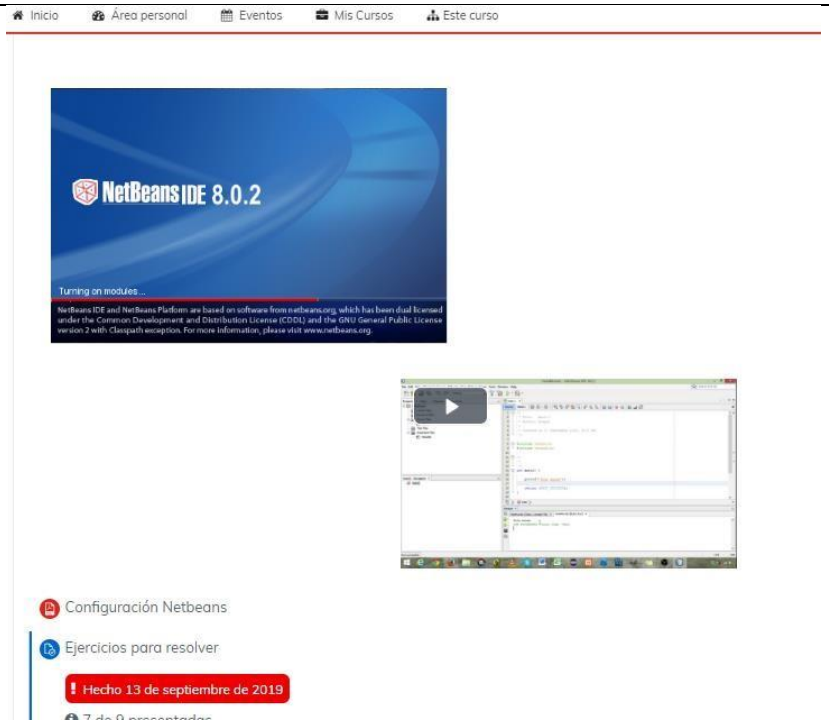
Prácticas en clase

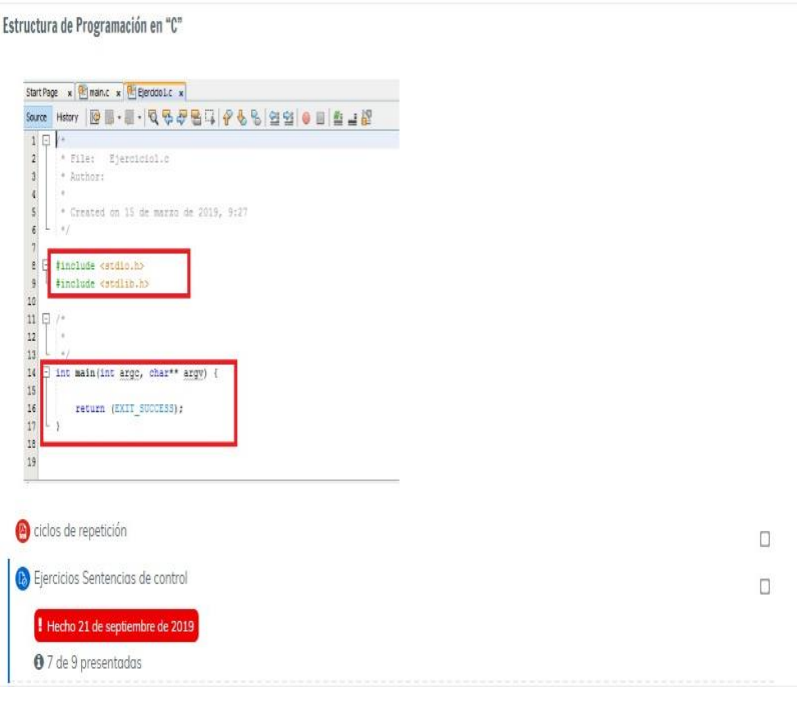




ANEXO 2

USO DE LA PLATAFORMA





Acción sobre las calificaciones Elegir...

Restablecer preferencias de tabla

Seleccionar	Imagen del usuario	Nombre / Apellido(s)	Dirección de correo	Estado	Calificación	Editar	Última modificación (entrega)	Archivos enviados	Comentarios de la entrega
		VINICIO MIGUEL CUEVA SAEZ	cuevavinicio1997@hotmail.com	Enviado para calificar Calificado	Calificación 7,50 / 10,00	Editar	sábado, 21 de septiembre de 2019, 18:49	Deber 2.docx	Comentarios (0)
		JHONY ALEXANDER DIAZ VASQUEZ	jhonydiaz1995@outlook.es	Enviado para calificar Calificado	Calificación 7,50 / 10,00	Editar	viernes, 20 de septiembre de 2019, 22:11	deber.pdf	Comentarios (0)
		LUIS RODRIGO PALAGUACHAY RUMIGUANO	lpalaguachay@gmail.com	Enviado para calificar Calificado	Calificación 7,00 / 10,00	Editar	viernes, 20 de septiembre de 2019, 19:55	deber clase2.pdf TRABAJO SEMANAL2.doc	Comentarios (1)
		LUIS ALBERTO TUQUERRES OYAGATA	luisis-065@hotmail.com	Enviado para calificar Calificado	Calificación 9,00 / 10,00	Editar	viernes, 20 de septiembre de 2019, 19:22	trabajo semanal 2 PDF.pdf	Comentarios (0)
		CECILIA ELIZABETH ROSALES VILLARREAL	cecyscta@yahoo.com	Enviado para calificar Calificado	Calificación 8,50 / 10,00	Editar	viernes, 20 de septiembre de 2019, 13:55	TRABAJO1 SEMANAL2para pdf.pdf	Comentarios (0)
		ROBERTO PAUL GUZMAN	roberto-thevipper@hotmail.com	Enviado para	Calificación	Editar	jueves, 19 de	TAREA #2.pdf	Comentarios

Arreglos y Estructuras

- Arreglos y Estructuras
- Deber N-3
  - ! Hecho 27 de septiembre de 2019
  - 7 de 9 presentadas, 7 sin clasificar
- Formato Proyecto Final
- Aviso Importante
- ManejoSwitch

FUNCIONES

INSTITUTO SUPERIOR TECNOLÓGICO

JAPÓN

Amor al conocimiento

Guía metodológica de lenguaje y programación
Carrera desarrollo de software
Mgs. Diana Moncayo
2019

Coordinación editorial general:
Mgs. Milton Altamirano Pazmiño
Ing. Alexis Benavides Vinueza
Mgs. Lucía Begnini Dominguez

Diagramación: Sebastián Gallardo Ramírez
Corrección de estilo: Mgs. Lucía Begnini Dominguez
Diseño: Sebastián Gallardo Ramírez
Imprenta: JKIMPRIMA

Instituto Superior Tecnológico Japón
AMOR AL CONOCIMIENTO

ISBN: 978-9942-811-83-7