UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

FACULTAD DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

CURSO

INTRODUCCIÓN A LA COMPUTACIÓN

GRUPO A

TRABAJO DE INVESTIGACIÓN FORMATIVA

Tema Del Lenguaje de Programación: C++

DOCENTE

Magister. Richart Escobedo Quispe

INTEGRANTES

-Melsy Melany Huamani Vargas
-Pedro Miguel Soncco Hancco
-Juan José Bruno Molleapaza Valdivia
-Diego Ivan Pacori Anccasi
-Edwin Francisco Aguilar Tancayo

CICLO 2021-I

ÍNDICE

-INTRODUCCIÓN AL LENGUAJE C++
-PARADIGMAS DE C++
-APLICACIONES DE C++
-VENTAJAS Y DESVENTAJAS DEL LENGUAJE C++
-CURVA DE APRENDIZAJE DE C++
-DIFERENCIAS Y SEMEJANZAS DEL LENGUAJE C CON EL LENGUAJE C++
-EJEMPLOS
-EJERCICIOS
-APLICACIÓN
-RECOMENDACIONES
-CONCLUSIONES
-REFERENCIAS

INTRODUCCIÓN AL LENGUAJE C++

HISTORIA DEL LENGUAJE C++

Para entender poco a poco este lenguaje comenzaremos por definirlo.

C ++ es un superconjunto creado a partir del lenguaje C, cuenta con todas las funcionalidades de C y agrega funcionalidades nuevas como clases, sobrecarga de funciones, herencia entre clases, etc.

-Origen

Su origen data del año 1979 y se le atribuye a Bjarne Stroustrup. El lenguaje que inspiró a Stroustrup fue el lenguaje Simula (lenguaje usado para simulaciones), que es considerado el primer lenguaje en permitir programación orientada a objetos. Stroustrup consideró que esta funcionalidad del lenguaje Simula era muy útil en el desarrollo de software, pero Simula era muy lento para un uso práctico.

Stroustrup comenzó a trabajar en su lenguaje llamado "C with classes" (C con clases) , su meta era agregar programación orientada a objetos al lenguaje C. El primer compilador de este lenguaje fue Cfront (un compilador escrito en C with classes) derivado del compilador de lenguaje C llamado CPre, aunque en 1993 se dejaría de usar por la dificultad para agregar nuevas funciones.

En 1983 el nombre del lenguaje fue cambiado de "C with classes" a "C++". Podemos entender con esto la imagen que tenía Stroustrup de su lenguaje como una mejora del lenguaje "C" (al ser ++ un incrementador de variable).

Para 1985 Stroustrup publicó su referencia al lenguaje "The C++ Programming Language" ("El lenguaje de programación C++), el cual fue muy importante debido a la falta de estandarización del lenguaje recién creado. En 1990 se publicó "The Annotated C++ Reference Manual" (Manual de referencia anotada de C++) y ese mismo año salió al mercado el compilador "Turbo C++" (desarrollado por Borland Software Corporation) que agregaba una gran cantidad de nuevas librerías al lenguaje, ayudando a su desarrollo. El proyecto de librerías "Boost" agregó nuevas funcionalidades al lenguaje tales como aleatorización comprehensiva y una nueva librería de tiempo.

-Estandarización

En 1998 el "Comité de estándares de C++" publicó su primera estandarización internacional ISO/IEC 14882:1998 (conocida también como C++98) la cual, al tener varios problemas, fue actualizada en 2003 (C++03). En 2011 se terminó y publicó la nueva estandarización del lenguaje (C++11).

PARADIGMAS DE C++

Comenzaremos definiendo lo que es un paradigma de un lenguaje de programación:

Paradigma de programación es una propuesta tecnológica que es adoptada por una Comunidad de Programadores cuyo núcleo central es incuestionable en cuanto a que unívocamente trata de resolver uno o varios problemas claramente delimitados. La resolución de estos problemas debe suponer consecuentemente un avance significativo en al menos un parámetro que afecte a la ingeniería de Software. Tiene una estrecha relación con la formalización de determinados lenguajes en su momento de definición. Un paradigma de programación está delimitado en el tiempo en cuanto a aceptación y uso ya que nuevos paradigmas aportan nuevas o mejores soluciones que la sustituyen parcial o totalmente.

-Ejemplo

Probablemente el paradigma de programación que actualmente es el más usado a todos los niveles es la orientación a Objeto. El núcleo central de este paradigma es la unión de datos y procesamiento en una entidad llamada "objeto", relacionable a su vez con otras entidades "objeto". Tradicionalmente datos y procesamiento se han separado en áreas diferente del diseño y la implementación de software. Esto provocó que grandes desarrollos tuvieran problemas de fiabilidad, mantenimiento, adaptación a los cambios y escalabilidad. Con la orientación a objetos y características como el encapsulado, polimorfismo o la Herencia se permitió un avance significativo en el desarrollo de software a cualquier escala de producción. La orientación a objeto parece estar ligado en sus orígenes con lenguajes como Lisp y Simula aunque el primero que le otorgó el titulo de programación orientada a objetos fue Smaltalk

-Tipos de Paradigmas de C++

- Programación Imperativa
- Programación funcional
- Programación lógica
- Declarativo
- POO Orientado a Objetos
- Por procedimientos

Habitualmente se mezclan todos los tipos de paradigmas a la hora de hacer la programación. De esa manera se origina la programación multiparadigma, pero el que actualmente es más usado de todos esos paradigmas es el de la programación orientada a objetos.

APLICACIONES DE C++

Las aplicaciones del lenguaje C++ son muy extensas. A continuación se muestran algunas:

Navegadores WEB: Utilizan C++ porque necesitan rapidez a la hora de mostrar los resultados en pantalla.

Sistemas operativos: La columna principal tanto de Windows, como Linux o Mac OS, están escritas en C++. Su potencia y rapidez lo hace un lenguaje de programación ideal para programar un sistema operativo.

Compiladores: los compiladores de muchos lenguajes de programación están escritos en C++.

Videojuegos: C++ es utilizado aún en el mundo de los videojuegos, bien para programar motores gráficos o para alguna parte concreta del videojuego.

Algoritmo 1: Solución para la llamada a un método sin conocer su declaración. InvocarMétodo es un método virtual de ClaseDinámica cuya declaración se conoce en la aplicación y MiMétodo es un método de Mi ClaseDinámica cuya declaración no se conoce en la aplicación

En la Aplicación

pObjeto->InvocarM´etodo (IDMétodo, param1, param2, ...)

En la Unidad de Extensión

MiClaseDinámica::InvocarMétodo (IDM´etodo, parametros)

Si (IDMétodo == IDMiMétodo)

param1 = RecuperarParámetro(parámetros, 1)

param2 = RecuperarParámetro(parámetros, 2)

…

MiM´etodo (parámetro1, parámetro2, ...)

Fin si

Fin InvocarMétodo

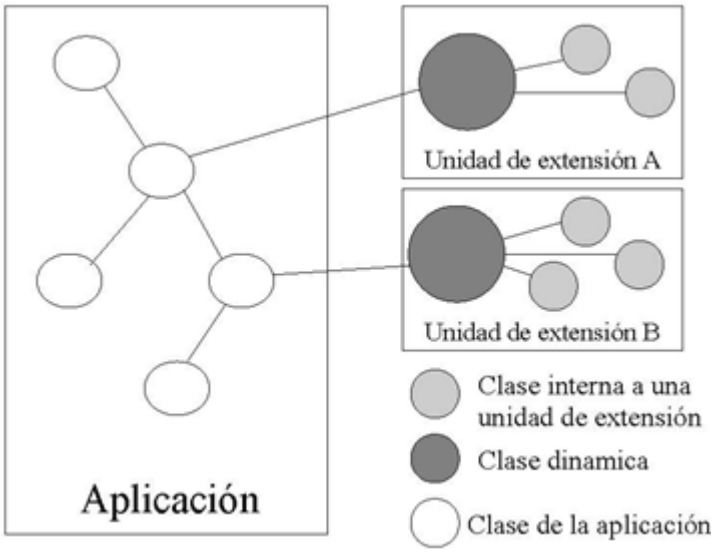
Algoritmo 2: Declaración de la clase A.

```
class A
{
public:
A () { //Inicializa la clase }
~A () { // Finaliza la clase }
unsigned M1 (int parameter1, float parameter2)
{ // Realiza algo }
};
```

Algoritmo 3: Implementación del método M2 de la clase B.

```
void B::M2
{
// ...
A* pA = newA();
//...
unsigned respuesta = pA->M1 (5, 2.6);
// ...
delete pA;
// ...
}
```

Unidades de extensión



Algoritmo 4: Implementación de la clase dinámica ClaseDinamicaA, la cual hereda de CDynamicClass. Se puede observar que la interacción de la aplicación con una clase dinámica se realiza mediante el método virtual CallMethod, el cual pertenece a CDynamicClass, y que se encarga de realizar la llamada al método deseado. Este método recibe como entradas un puntero al objeto sobre el cual el usuario desea realizar una acción, un identificador del método a llamar y una lista de los parámetros del método. El constructor de la clase debe declarar todos los métodos de la clase dinámica que pueden ser accedidos desde fuera de la unidad de extensión a la que pertenece.

```
#define CA M1 1000001
ClaseDinamicaA::ClaseDinamicaA (CClassInfo* theClassInfo): CDynamicClass (theClassInfo)
{
CMethodSpecification* pMethod;
pMethod = RegisterMethod (CA  M1, "M1", "Un m´etodo", CMethodSpecification::cUINT, 2);
pMethod->AddParameter (CMethodSpecification::cInt, "parameter 1", 1, "int parameter1");
pMethod->AddParameter (CMethodSpecification::cFloat, "parameter 2", 2, "float parameter2");
}
CMultivalueObject ClaseDinamicaA::CallMethod(void* *pObject, unsigned iMethodID, va_list pListOfParameters)
{
CMultivalueObject result; InitCriticalSection (); if (iMethodID == CA _M1)
{
int parameter1 = va_arg (pListOfParameters, int); float parameter2 = va_arg (pListOfParameters, float);
result = CMultivalueObject (((A*)pObject)->M1(parameter1, parameter2));
}
FinishCriticalSection ();
return result;
}
void* ClaseDinamicaA::GetNewObject()
{
return new A ();
}
void ClaseDinamicaA::FinishObject(void *theObject)
{
delete (A*) theObject;
}
```

Algoritmo 5 : Implementación de los métodos GetClassID que devuelve el código identificador de la clase, GetDescription que retorna una cadena descriptiva de la clase, Init que crea un objeto de la clase dinámica y retorna su dirección, y Finish que destruye el objeto de la clase dinámica.

```
#include "ClaseDinamicaA.h"
ClaseDinamicaA *pClaseDinamica = NULL;
extern "C" int APIENTRY
DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID lpReserved)
{
return 1; // ok
}
extern "C" void* WINAPI Init (void* classInfoP)
{
pClaseDinamica = new ClaseDinamicaA ((CClassInfo*)classInfoP); return pClaseDinamica;
}
extern "C" char* WINAPI GetDescription()
{
return "Clase A";
}
extern "C" CLSID WINAPI GetClassID()
{
/* Vendor ID: 34E300C1 Oscar Antezana Chavez
Class Type: FFEE Clases de Prueba
Class ID: 1001 CA
Version: 01 Primera versi´on p´ublica
Type: 01 Primera versi´on interna
Key Value: 00000000 Sin compatibilidad
Passwd: 0000 Sin clave */
// ID: 34E300C1-FFEE-1001-0101-00000000000
CLSID clsidID = 0x34e300c1, 0xFFEE, 0x1001,
0x01, 0x01, 0x0, 0x0, 0x0, 0x10, 0x0, 0x0 ;
return clsidID;
}
extern "C" void WINAPI Finish ()
{
delete pClaseDinamica;
}
```

VENTAJAS Y DESVENTAJAS DEL LENGUAJE C++

VENTAJAS:

- **Alto rendimiento:**

  El alto rendimiento que ofrece es debido a la eficiencia que posee cuando se realizan llamadas directas al sistema operativo, puesto que es un lenguaje compilado para cada plataforma ofreciendo una gran variedad de parámetros de programación e incluso se acopla de manera efectiva con el lenguaje ensamblador. Además esto permite usar eficazmente la memoria y los procesos suelen realizarse con mayor velocidad a comparación de otros lenguajes de programación.

- **Multiplataforma:**

  El lenguaje de programación C++ es considerado un lenguaje híbrido, es decir, un lenguaje de nivel medio ya que puede modificar el hardware de una máquina (bajo nivel) pero también puede crear órdenes independientes (alto nivel). Entonces decimos que C++ es un lenguaje multiplataforma, pues permite ejecutar sus programas en cualquier software o hardware, y sus aplicaciones funcionan en cualquier plataforma.

- **Extendido:**

  Se dice que es extendido porque la mayoría de programas o parte de los programas están realizados en base a este lenguaje. Además, permite desarrollar sistemas operativos (Windows, Linux, Mac, Android), utilidades (bibliotecas, servicios, mantenimiento), software de desarrollo (compiladores, depuradores, IDE), entre otras funciones. Asimismo cabe mencionar que las grandes empresas suelen utilizar este lenguaje para crear aplicaciones móviles que funcionen en navegadores web.

- **Gestor de base de datos:**

  La gran parte de los gestores de datos como MySQL o PostgreSQL están programados en este lenguaje, esto se debe a que es un elemento importante de C++ que permite garantizar el intercambio, almacenamiento o actualización de datos.

- **Versatilidad con el backend:**

  Este lenguaje de programación ha sido de utilidad para crear los compiladores de los diversos lenguajes, por ello posee mayor versatilidad al programar al lado del backend. Aprovechando ese factor, se han desarrollado con mayor frecuencia aplicaciones bancarias con C++.

- **Creación de gráficos:**

  El lenguaje de programación C++ permite diseñar programas que aceleran el proceso de imágenes para elaborar gráficos relacionados a estadísticas. Además permite que los programas puedan ser ejecutados en cualquier sistema, por esa razón es muy utilizado por la industria de juegos online o de consolas.

- **Multiparadigma:**

  Tal y como el nombre lo dice y recalcando lo que se describió con anterioridad, C++ puede soportar todos los paradigmas de programación o la mayoría de ellos, por ejemplo tenemos: orientado a objetos, orientado a aspectos, modular, lógico, imperativo, funcional, declarativo, estructurado, entre otros. Siendo este último y el orientado a objetos los más utilizados o especializados en este lenguaje.

- **Actualizado:**

  El lenguaje C++ posee una gran comunidad que actualiza de manera constante las librerías de C++, es por ello, que es factible la reutilización de código en los programas que desarrollemos. Además permite operar con datos complejos e implementar una variedad de patrones de diseño.

DESVENTAJAS:

- **Depuración complicada:**

  Al momento de la codificación del programa tiene un gran fallo en la depuración de errores, tomando mucho tiempo para solucionarlos, esto sucede aun cuando C++ ha mejorado en relación al lenguaje C.

- **Traducción compleja al lenguaje máquina:**
  El lenguaje C++ no facilita con el uso de sus operadores la traducción eficiente al lenguaje máquina por lo que suele terminar siendo muy compleja al momento de evaluar, además no ayuda a realizar las operaciones más abstractas del programa.

- **No recomendable para sitios web:**
  Si bien es cierto muchos programadores han optado por utilizar el lenguaje C++ para crear sitios web no es muy recomendable, puesto que no está familiarizado con programas del lado de frontend.

- **Poca preferencia:**
  Aunque el lenguaje C++ es multiparadigma y puede ser utilizado en cualquier función que se disponga, sin embargo como el lenguaje en sí es complicado de utilizar los programadores optan por otros lenguajes de programación especializados en lo que necesiten, dejando de lado el lenguaje C++.

- **Complejidad en el uso de DLLs:**
  A diferencia de otros lenguajes que mejoraron en la manipulación de DLLs mediante sus frameworks, en C++ el desarrollador debe ser capaz de cargar y liberar las librerías dinámicas de la memoria corriendo riesgos por el manejo de esta. Sin embargo, por la inexperiencia o falta de costumbre de los programadores puede llevar a generar errores en el programa, lo que es difícil de solucionar, además si se olvida de liberar la memoria, esta termina ocupada al finalizar el programa.

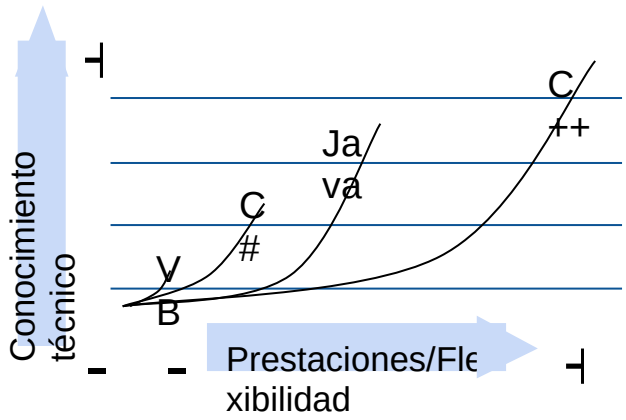
- **Interfaz poco atractiva:**
  A pesar de su gran funcionalidad, el lenguaje de programación C++ no posee una interfaz visualmente atractiva para los programadores por lo que al principio suele ser complicado poder entender cómo funciona.

- **Curva de aprendizaje alta:**
  En general, el lenguaje C++ es muy complicado para que un nuevo programador comience a aprender, ya que primero se debe conocer el lenguaje C además de saber programarlo. En adición a ello, se requiere un estudio profundo a la sintaxis, palabras, estructuras, etc; por lo que tomará mayor tiempo el dominar este lenguaje.

CURVA DE APRENDIZAJE DE C++

Una curva de aprendizaje es cuando se consideran dos factores necesarios para el adquirir conocimientos del tema que se está estudiando. En este caso, consideramos el aprendizaje de distintos lenguajes de programación incluyendo el de C++ como nuestro tema principal, mientras que los factores serán el conocimiento técnico que se necesita para trabajar con ello y las prestaciones o flexibilidad que ofrece para tener un mejor aprendizaje del lenguaje.

De ello se pudo obtener el siguiente gráfico descriptivo:

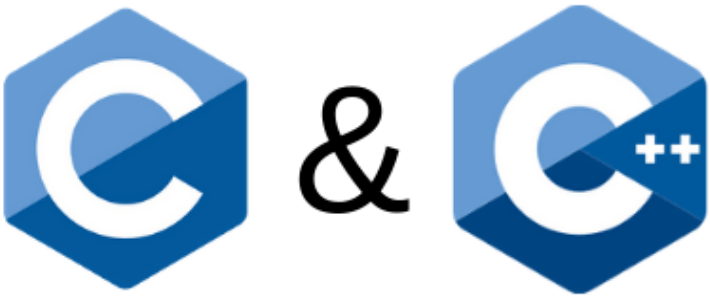


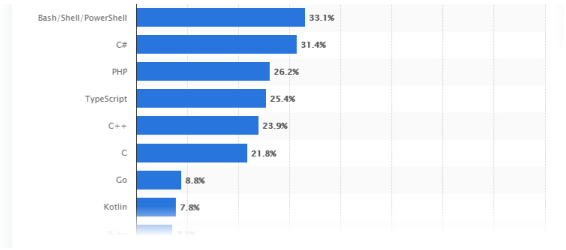
Como se puede apreciar y como se ha mencionado con anterioridad, uno de los aspectos negativos de este lenguaje es su alta curva de aprendizaje, ya que suele ser muy complicado para que nuevos programadores aprendan sus sintaxis, palabras o funciones que derivan del lenguaje C y los propios del C++. Además al considerar el fallo de la depuración de errores, lo que provoca una inadecuada solución y la complejidad de su uso de DLLs, ya que no se optimizó su manipulación por medio frameworks como en otros lenguajes. Entonces el lenguaje C++, se podría describir como un lenguaje muy complicado, es por ello que en los

últimos años su uso ha sido muy precario a comparación de otros lenguajes con mayor preferencia.

DIFERENCIAS Y SEMEJANZAS DEL LENGUAJE C CON EL LENGUAJE C++

Siempre ha existido cierta duda en la diferencia de ambos lenguajes y es que por su historia ambos se encuentran relacionados, sin embargo, esto no quiere decir que sean igual para el desarrollo de programas.





Una de las diferencias que destacan más entre C y C++ es que el segundo es un lenguaje de programación orientado a objetos, por lo que permite el encapsulamiento de información y solo objetos que tengan acceso podrán modificar las clases creadas. Distinto al lenguaje C, donde no existen las clases por lo que no existe un ocultamiento de información y todo está escrito en un mismo código para el funcionamiento y la realización del programa.

Entonces, se puede decir que en el lenguaje C no existe la herencia de clases, que consiste en que una superclase transmitirá sus funciones y atributos a las clases hijas (o también denominadas subclases), mientras que en el lenguaje C++ como sí es un lenguaje de programación orientado a objetos permite que las subclases adopten los atributos necesarios de las superclases.

De lo anterior podemos alegar sobre el uso del polimorfismo en el lenguaje de programación C++, ya que al tener varios objetos diferentes creados en cada clase, se puede enviar una misma

acción a los objetos sin necesidad de que sean de la misma clase, esto ocurre totalmente contrario a lo que sucedería en el caso del lenguaje C.

Es importante mencionar que, mientras que C++ opta por trabajar con un paradigma de programación estructurado, en el lenguaje C se usa para una programación basada en procedimientos; este tipo de paradigma se define como un método que no utiliza repeticiones, sino que siempre se debe expresar la función que se va a llamar.
Otro factor necesario para destacar es el hecho de que en el lenguaje C las líneas de código son mucho más largas a comparación del lenguaje C++, por lo que esto dificulta aún más la rápida solución de los errores de código.

Por otro lado, en C++ se pueden utilizar características provenientes del lenguaje C como punteros básicos, es decir, variables que respalden las direcciones de la memoria, además pueden trabajar con matrices y cadenas de caracteres finalizadas en Null (no existe valor).

Por consiguiente, para la solución de fallos en un programa realizado en el lenguaje de programación C++ se usan las excepciones para crear un código de identificación de errores. Si es que se diera el caso que el error se repitiera en cada ejecución el elimina el código general para no afectar a la continuidad del desarrollo.

Otra de las diferencias entre C y C++ es que a pesar de que los dos lenguajes utilizan tokens para su código (palabras y signos de puntuación de la lengua humana), en C solo se encuentran 32 tipos, mientras que en C++ se añadieron 31 más llegando a la totalidad de 63 tokens para la creación de código.

Por último, la diferencia entre sus extensiones (.c y .cpp):

Programa .c	Programa .cpp
Esta extensión hace referencia a los archivos que se encuentran en el lenguaje de programación C. Restringe al compilador para que solo el	Esta extensión hace referencia a los archivos que se encuentran realizados en el lenguaje C++. El compilador es mucho más flexible por lo

lenguaje C lo pueda emplear mas no permite a los programas realizados en C++.	que permite emplear en los programas de C y en los programas de C++. Por otro lado si no se especifica la extensión, por defecto tomará .cpp.

EJERCICIOS

1. Básico

1.1. Hola Mundo

holamundo.cpp | Impresion.cpp | temporal.cpp

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

int main(){

    cout<<"Hola Mundo"<<endl;
    return 0;
}
```

```
D:\C++ trabajos\Hola Mundo\holamundo.exe
Hola Mundo

--------------------------------
Process exited after 0.3129 seconds with return value 0
Presione una tecla para continuar . . .
```

1.2. Condiciones

```cpp
#include <iostream>
// Cómo saber si un triángulo existe y saber si es isósceles, escanelo o equilátero
using namespace std;
int main(){
    double lado1,lado2,lado3;
    cout<<"Ingrese los 3 lados del triangulo"<<endl;
    cin>>lado1;
    cin>>lado2;
    cin>>lado3;
    if(lado1+lado2>lado3 && lado1+lado3>lado2 && lado2+lado3>lado1){
        cout<<"Este triangulo existe"<<endl;
        if(lado1==lado2||lado2==lado3||lado3==lado1){
            cout<<"Es un triangulo isosceles"<<endl;
        }
        else if(lado1==lado2 && lado2==lado3){
            cout<<"Es un triangulo equilatero"<<endl;
        }
        else if(lado1!=lado2 && lado2!=lado3){
            cout<<"Es un triangulo escaleno"<<endl;
        }
    }
    else{
        cout<<"Este triangulo no existe"<<endl;
    }
    return 0;
}
```

```
D:\C++ trabajos\Triangulos.exe
Ingrese los 3 lados del triangulo
5
6
7
Este triangulo existe
Es un triangulo escaleno

--------------------------------
Process exited after 17.38 seconds with return value 0
Presione una tecla para continuar . . .
```

2. Intermedio

2.1. Repetición y Arrays

holamundo.cpp | Impresion.cpp | temporal.cpp

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

int main()
{
    cout<<"Ingrese el tamaÃ±o del arreglo"<<endl;
    int x;
    cin>>x;
    int arreglo[x];
    cout<<"Ingrese los valores:"<<endl;
    for(int i=0;i<x;i++)
    {
        cin>>arreglo[i];
    }

    for(int y=0;y<x;y++)
    {
        cout<<arreglo[y]<<" ";

    }

    return 0;
}
```

```
D:\C++ trabajos\Impresion\Impresion.exe
Ingrese el tamaño del arreglo
5
Ingrese los valores:
10
11
12
13
14
10 11 12 13 14
--------------------------------
Process exited after 5.117 seconds with return value 0
Presione una tecla para continuar . . .
```

2.2. Método, array y repetición

```cpp
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
//Problema:
//Simular un lanzamiento de n dados para m concursantes, indicar el ganador con su respectivo puntaje
using namespace std;
int Dado(int a);
int main(){
    int n,m,dado1;
    cout<<"Ingrese número de dados a lanzar: ";   cin>>n;
    cout<<"Ingrese número de personas a jugar: "; cin>>m;
    int per[m];
    for(int i=0;i<m;i++){
        per[i]=Dado(n);
    }
    int max=0;
    for(int j=0;j<m;j++){
        cout<<"Player "<<j+1<<"---->"<<per[j]<<endl;
        if(per[j]>=max){
            max=per[j];
        }
        else if(per[j]<=max){
            max=max;
        }
    }
    int contador=0;
    cout<<"Los Ganadores son: "<<endl;
    for(int k=0;k<m;k++){
        if(per[k]==max){
            cout<<"El player "<<k+1<<" "<<"obtuvo el mayor puntaje: "<<max<<endl;
            contador++;
        }
        else{
        }
    }
    return 0;
}
int Dado(int a){
    int suma=0;
    for(int i=0;i<a;i++){
        int alea=rand()%(6-1+1)+1;
        suma=suma+alea;
        alea=0;
    }
    return suma;
}
```

```
D:\C++ trabajos\intermedio2.exe
Ingrese n·mero de dados a lanzar: 2
Ingrese n·mero de personas a jugar: 25
Player 1---->12
Player 2---->10
Player 3---->11
Player 4---->2
Player 5---->8
Player 6---->12
Player 7---->6
Player 8---->8
Player 9---->5
Player 10---->5
Player 11---->5
Player 12---->7
Player 13---->10
Player 14---->7
Player 15---->9
Player 16---->7
Player 17---->7
Player 18---->9
Player 19---->8
Player 20---->3
Player 21---->10
Player 22---->7
Player 23---->7
Player 24---->6
Player 25---->11
Los Ganadores son:
El player 1 obtuvo el mayor puntaje: 12
El player 6 obtuvo el mayor puntaje: 12

--------------------------------
Process exited after 2.535 seconds with return value 0
Presione una tecla para continuar . . .
```

3. Avanzado

holamundo.cpp | Impresion.cpp | [*] temporal.cpp

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

int main()
{
    cout<<"numero de elementos"<<endl;
    int x;
    cin>>x;
    int arreglo[x];
    for(int i=0;i<x;i++)
    {
    cin>>arreglo[i];
    }
    int *puntero;
    puntero=arreglo;

    //utilizado para imprimir valores de un arreglo

    for(int j=0;j<x;j++)
    {
        cout<<"Elemento ["<<j<<"]"<<*puntero++<<endl;
    }


    //puntero usado para imprimir direcciones de memoria de un arreglo

    for(int k=0;k<x;k++)
        {
            cout<<"Posicion de memoria ["<<k<<"]"<<puntero++<<endl;
        }
}
```

```
D:\C++ trabajos\Metodo\temporal.exe
numero de elementos
6
1
2
3
4
5
6
Elemento [0]1
Elemento [1]2
Elemento [2]3
Elemento [3]4
Elemento [4]5
Elemento [5]6
Posicion de memoria [0]0x6ffdb8
Posicion de memoria [1]0x6ffdbc
Posicion de memoria [2]0x6ffdc0
Posicion de memoria [3]0x6ffdc4
Posicion de memoria [4]0x6ffdc8
Posicion de memoria [5]0x6ffdcc

--------------------------------
Process exited after 6.129 seconds with return value 0
Presione una tecla para continuar . . .
```

APLICACIÓN

Base de Datos: MySQL, una de las bases de datos más usadas está escrita en C++.

Navegadores WEB: Nos muestran más rápido los resultados en pantalla.

Sistemas Operativos: La columna principal tanto de Windows, Linux y Mac OS, están hechas en C++. Debido a su potencia y rapidez lo hace ideal para un sistema operativo.

Compiladores: Muchos compiladores de otros lenguajes están hechos en C++.

Videojuegos: Muchas veces es utilizado en motores gráficos o en alguna parte específica del videojuego. También tiene otras aplicaciones como en máquinas médicas, relojes inteligentes, etc. Por su capacidad de estar cerca del lenguaje máquina que otros lenguajes de alto nivel.

C++ es tan poderoso que se utiliza en muchos proyectos importantes: TensorFlow para Machine Learning, V8 como el motor de JavaScript para Google Chrome y Node.js, Electron para crear aplicaciones de escritorio con HTML, CSS y JavaScript, entre otras.

Además, muchos motores de Videojuegos como **Unreal Engine**, Creation Engine, CryEngine o Source usan C++. De hecho, ¿me crees si te digo que **Unity 3D** también utiliza C++? Si, programamos en C#. Pero Unity lo compila todo a C++.

RECOMENDACIONES

Aunque son los lenguajes con los que la mayoría de las personas comienzan a programar (C y C++). Aprender cualquiera de ellos te puede servir mucho en tu vida profesional puesto que ambos lenguajes son rápidos en ejecución, te permiten tener control sobre las tareas que realizas, son escalables y su documentación es muy grande; lo anterior se traduce en una alta probabilidad de encontrar ofertas de trabajo en las que busquen programadores con conocimientos sobre C o C++.

La mayoría de la gente no usa C++ de la manera que podría ser usado, es decir C++ contiene muchos elementos que podrían se optimizados para crear programas y aplicaciones más eficientes.

Muchos siguen tratando de usar C++ como una versión mejorada de C, cuando esa no es la forma correcta de usarlo.

Una vez se entienda la lógica de la programación, la mejor recomendación para aquellos que quieran aprender C++ (o cualquier otro lenguaje) de la manera más eficiente es leer un buen libro sobre el tema, y así aprender un poco sobre el lenguaje en cuestión, incluir historia, fundamentos, investigar los avances que dicho lenguaje ha tenido y tratar de entender por qué se creó, cuál es su propósito, qué se puede llegar a crear, etc.

CONCLUSIONES

Este Trabajo de Investigación Formativa presenta el diseño y la implementación de un lenguaje de programación que elimina los problemas léxicos, sintácticos y semánticos de C++ y provee un adecuado soporte para la programación genérica, la cual tiene por objetivo lograr la implementación eficiente de algoritmos a diferencia de otros paradigmas de programación que consideran el problema de expresar algoritmos como resuelto y que por lo mismo están dirigidos al modelado y diseño de software.

En este Trabajo de Investigación Formativa se muestra cómo al intentar cumplir los postulados de la programación genérica se hacen evidentes las carencias de muchos lenguajes de programación en cuanto a su capacidad de expresar algoritmos eficientes en términos generales. También se explica cómo C++ intentó eliminar sin éxito sus debilidades en cuanto a su soporte para este paradigma. Se muestra cómo el lenguaje presentado resuelve todas las debilidades encontradas en C++.

Además de presentar el diseño de algunas características especialmente diseñadas para resolver algunas necesidades de la programación genérica se diseñó: una serie de ejercicios para facilitar la comprensión y aplicación de C++ en los estudiantes de Ingeniería que lo requieran.

La implementación del lenguaje presentada en este Trabajo de Investigación Formativa es un intérprete que realiza el proceso de compilación eficientemente lo cual es necesario para poder sobrellevar las dificultades causadas por la programación genérica en cuanto a la factibilidad de realizar compilación separada.

Además, la implementación permite ejecutar algunos programas mal formados de manera similar a los intérpretes de lenguajes dinámicos.

Lo más importante es que la implementación logra demostrar la factibilidad del lenguaje presentado, así como un análisis profundo desde.su historia hasta sus aplicaciones.

REFERENCIAS

[1] O. Antezana Chavez y G. Argote García, “Desarrollo de Aplicaciones C++ Modificables o Extensibles en Tiempo de Ejecución”, 2021. Disponible: http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S1683-07892001000200003

[2] J. Llort, “Diferencia entre C y C++, e introducción a las funciones”, Nov. 29, 2019. Disponible: https://cipsa.net/diferencia-c-cplus-introduccion-funciones/

[3] C. Herrera, “Ventajas y desventajas de usar C++ en la programación web”, Nov. 27, 2020. Disponible: https://blogueropro.com/blog/ventajas-y-desventajas-de-usar-c-en-la-programacion-web

[4] A. Robledano, “Qué es C++: Características y aplicaciones”, Jul. 22, 2019. Disponible: https://openwebinars.net/blog/que-es-cpp/