EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE IZTAPALAPA
TECNM MEXICO

INSTITUTO TECNOLOGICO DE IZTAPALAPA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

"libc++abi" C++ Standard Library Support

Presenta:	**No. Control**
Arzate Cervantes Alejandro	**171080098**
Hernandez Sanchez Ismael	**171080133**
Medina Piña José Tenatic	**171080116**
Rojas Martinez Yorman Jesus	**161080213**

Asesor:

CIUDAD DE MÉXICO NOVIEMBRE/2020 ,

Resumen

En el presente proyecto se realizará una investigación para la futura implementación de la librería libc++abi y la relación que tiene con el funcionamiento del proyecto LLVM. El proyecto se basa en una investigación a fondo de dicha librería, conocerla a fondo para tener una visión general de ella para demostrar de manera práctica el desarrollo de un ejemplo en funcionamiento del mismo, así como una demostración de cómo podemos modificar la estructura de la misma para cambiar el funcionamiento de la librería en caso de que el usuario necesite una característica en especial.

Cronograma de Actividades

S E M A N A S

ACTIVIDAD	1	2	3	4	5	6	7	8	9	10	11	12	13	14
INVESTIGACIÓN DE LLVM														
INVESTIGACIÓN DE LIBC++ABI														
SÍNTESIS DE LA INFORMACIÓN														
IMPLEMENTACIÓN DE LLVM														
IMPLENTACION DE LIBC++ABI														
PRUEBAS														
PRESENTACION DEL PROGRAMA														
OBSERVACIONES														

Análisis de Riesgo

Riesgos	***Soluciones Propuestas***
desbordamiento de bufer ***Apple Mac OS X hasta 10.11.5 (Operating System).*** *Esto tiene repercusión sobre la confidencialidad, integridad y disponibilidad.*	*Una actualización a la versión 10.11.6 elimina esta vulnerabilidad.*
c ++ 11 requieren la nueva implementación libc ++ de la biblioteca estándar de C ++. Pero libc ++ no es compatible con ABI con el antiguo libstdc ++	*algunas actualizaciones en librerías de c++ carecen de actualizaciones capaces de ser compatibles con otras versiones de ABI*
todas mis bibliotecas instaladas con macports tienen una ABI diferente para las clases std como cadena y no se pueden vincular con proyectos que hacen un uso intensivo de c ++ 11	*libc ++ también garantiza que las funciones de bajo nivel como las excepciones y la asignación de memoria dinámica sean compatibles con ABI (suponiendo que compile libstdc ++ y libc ++ usando la misma biblioteca abi)*

Libc++ tiene como objetivo preservar la ABI estable para evitar errores sutiles cuando el código creado en la antigua ABI está vinculado con el código creado en la nueva ABI. Al mismo tiempo, libc ++ permite mejoras que rompen ABI y correcciones de errores para los escenarios cuando el cambio de ABI no es un problema.

Para admitir ambos casos, libc ++ permite especificar la versión ABI en el momento de la compilación. La versión se define con una opción de cmake LIBCXX_ABI_VERSION. Otra opción LIBCXX_ABI_UNSTABLE se puede utilizar para incluir todas las características de ruptura ABI presentes. Estas opciones se traducen en definiciones de macro de C ++ _LIBCPP_ABI_VERSION, _LIBCPP_ABI_UNSTABLE.

Cualquier característica de cambio de ABI se coloca en su propia macro, _LIBCPP_ABI_XXX, que se habilita en función del valor de _LIBCPP_ABI_VERSION. _LIBCPP_ABI_UNSTABLE, si está configurado, habilita todas las funciones a la vez.

la implementación de la librería libc++abi y la relación que tiene con el funcionamiento del proyecto LLVM es la modificación de las funciones que pertenecen a la librería permitirán

- La corrección según lo definido por el estándar C ++ 11.
- Una proporción en una subcapa portátil para facilitar la migración de libc ++
- En Mac OS X, sea compatible con ABI con el soporte de bajo nivel existente.

refiriéndonos al soporte de bajo nivel para la comunicación estándar de c++; los paquetes de libc++ abi están destinados a proporcionar ciertas funciones de bajo nivel las cuales proveen una construcción de

la biblioteca estándar de c++ la cual es utilizada como soporte o apoyo como un componente separado para c++

Justificación

Las bibliotecas ABI o Interfaces Binarias de Aplicación contienen soporte para la administración de memoria, control de excepciones, etc, y están diseñadas para proporcionar ciertas funciones de bajo nivel a partir de las cuales se va a compilar la biblioteca estándar de C++.
Tenemos como objetivo demostrar de manera muy simple pero entendible el funcionamiento de esta librería que permite una mejor compatibilidad con el ejemplo de un buen diseño de un proyecto de software, estos se basan en módulos. Los módulos, los cuales en un proyecto hacen el papel de auxiliares en la metodología del diseño en bajo nivel de un programa, permitiendo definir por separado las distintas partes de un programa y las interfaces entre ellas. Además, un módulo permite agrupar elementos que compartan alguna afinidad.
Entonces, se tiene como ejemplo un programa que realiza el registro de personas, vamos a manejar dos conceptos básicos en esta librería que son el tratamiento de excepciones y la administración de memoria, esto a partir de una lista de personas con ciertos datos y que el registro o consulta de los mismos trabajara en bajo nivel errores que puedan llegarse a dar y que serán tratados con excepciones y tambien funcionara con el uso de interfaces, esto para demostrar la modularidad de un programa y las ventajas que nos da de compatibilidad que tenemos con diferentes tipos de objetos, todo esto llevado a cabo bajo el lenguaje de programación Java.

Referencias:
https://opensource.apple.com/source/gcc/gcc-5026.1/libstdc++-v3/docs/html/abi.html
https://informaticapc.com/poo/interfaces.php
https://libcxxabi.llvm.org/
https://stackoverflow.com/questions/45314177/what-is-the-difference-between-libc-and-libcabi-library-in-llvm
http://www.lcc.uma.es/~afdez/apuntes/laboratorio/apuntes/modulos.pdf

Metodología

Utilizando una metodología ágil **SCRUM** ya que tiene una duración fija y corta , con nuestro objetivo definido el cual es el uso de la librería libc++abi y delimitando el tiempo con el cierre del semestre, por lo general este método no cuenta con un líder, pero el equipo como unidad decidirá cómo abordar el problema y resolverlos, cada integrante formará parte integral para la solución y todos los integrantes desempeñamos los roles que se describen tanto como el dueño del producto en este caso seria el docente que solicita la implementación del proyecto de llvm; el ScrumMaster el cual sería el líder del equipo; y por último el equipo de desarrollo que de forma integral todos los integrantes del equipo son los participantes

El uso y la implementación de sprint (el periodo de tiempo donde se completa el trabajo específico) serán determinados en los Planning Sprint o reuniones del equipo asi ves entre mas se hacer que el momento de entregas se utilizarán los stand-up o reuniones diarias y breves

Los dos principales puntos por el que se escoje esta metodología son:

Las reglas de Scrum ágil dependen completamente del equipo y regirse por lo que funciona mejor para sus procesos, de forma sencilla podemos asignar los eventos básicos del scrum y las inspecciones y se adaptará según las necesidades únicas del equipo

El segundo punto es que Scrum es ampliamente utilizado por los equipos de software. es el método ágil más popular, según el 12ª informe anual de State of Agile, el 70% de los equipos de software usan Scrum o un híbrido Scrum. Por lo tanto el familiarizarnos con una metodología de trabajo ampliamente utilizada en este ámbito de trabajo nos dará un aporte para futuras referencias

Requerimientos Funcionales

El NDK(Kit de desarrollo nativo) admite varias bibliotecas de tiempo de ejecución C++.

libc++ de LLVM es la biblioteca estándar de C++ que usa el SO Android desde la versión Lollipop y, a partir del NDK r18, es la única STL disponible en el NDK.

El entorno de ejecución de C++ del sistema ofrece compatibilidad con la ABI básica del entorno de ejecución de C++. Básicamente, esta biblioteca proporciona `new` y `delete`. A diferencia de las otras opciones disponibles en el NDK, no se admite el manejo de excepciones o RTTI.

Requerimientos No Funcionales

Tiempos de ejecución estáticos

Si todo el código nativo de la aplicación está contenido en una única biblioteca compartida, te recomendamos usar el tiempo de ejecución estático. De esta manera, el vinculador podrá intercalar y recortar la máxima cantidad de código sin utilizar, de modo que el resultado sea una aplicación lo más pequeña y optimizada posible. Asimismo, evita los errores de PackageManager y del vinculador dinámico en las versiones anteriores de Android que dificultan el manejo de múltiples bibliotecas compartidas y facilitan la aparición de errores.

Dicho esto, en C++, no es seguro definir más de una copia de la misma función o del mismo objeto en un único programa. Este es un aspecto de la Regla de definición única (ODR) presente en el lenguaje C++ estándar.

Cuando se usa un tiempo de ejecución estático (y, en general, bibliotecas estáticas), esta regla puede infringirse por accidente fácilmente. Por ejemplo, la siguiente aplicación infringe esta regla:

En este caso, la STL (incluidos los constructores estáticos y los datos globales) está presente en ambas bibliotecas. El comportamiento de tiempo de ejecución de esta aplicación no está definido, por lo que las fallas son muy frecuentes en la práctica. Entre otros posibles problemas se incluyen los siguientes:

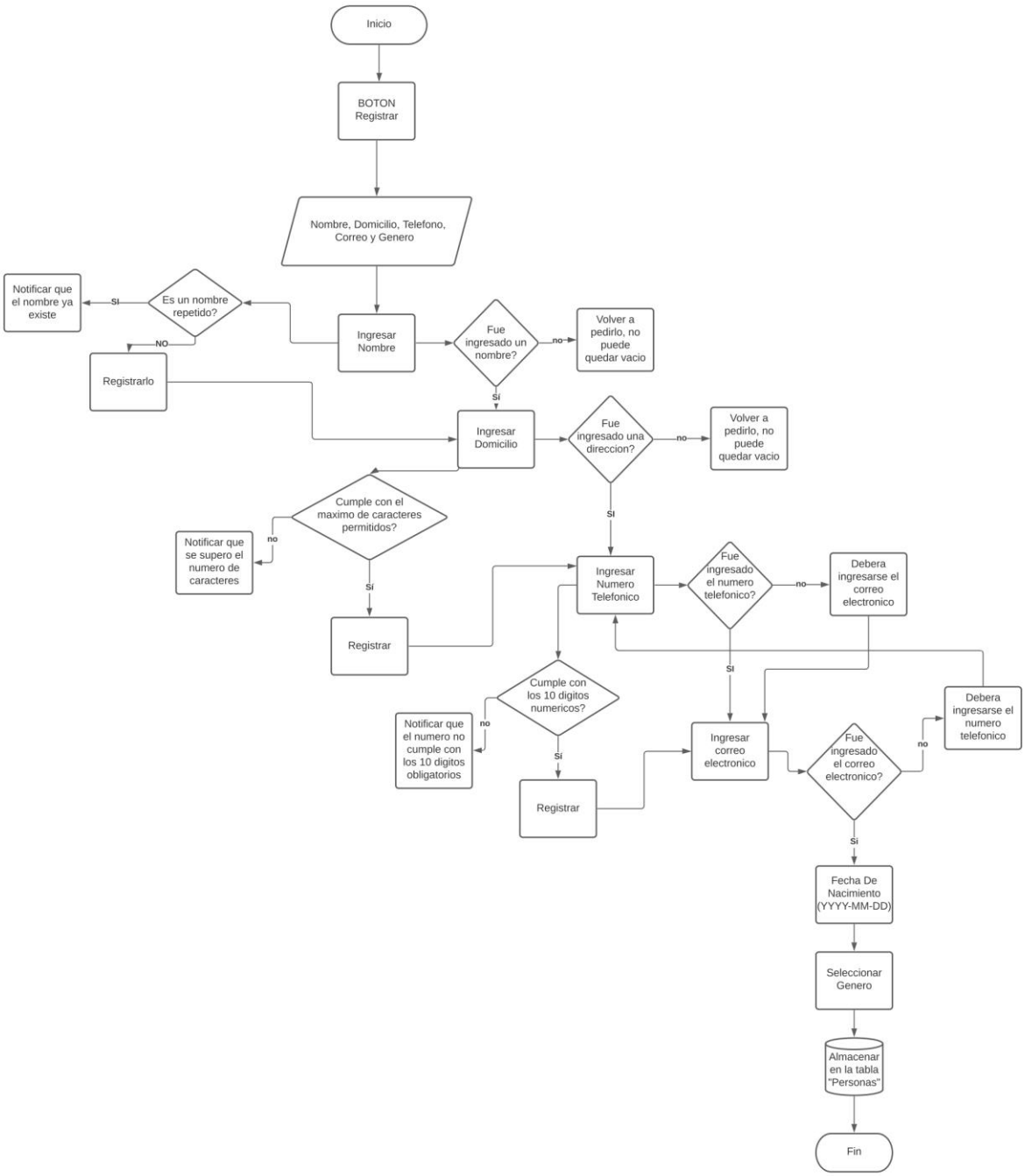
- Memoria asignada a una biblioteca, y liberada en la otra, lo que provoca fugas de memoria o daños en la pila.
- Las excepciones que surgen en `libfoo.so` no se detectan en `libbar.so` y provocan fallas en tu app.
- El almacenamiento en búfer de `std::cout` no funciona correctamente.

Más allá de los problemas de comportamiento, la vinculación del tiempo de ejecución estático con múltiples bibliotecas duplica el código en cada biblioteca compartida, lo que incrementa el tamaño de la aplicación.

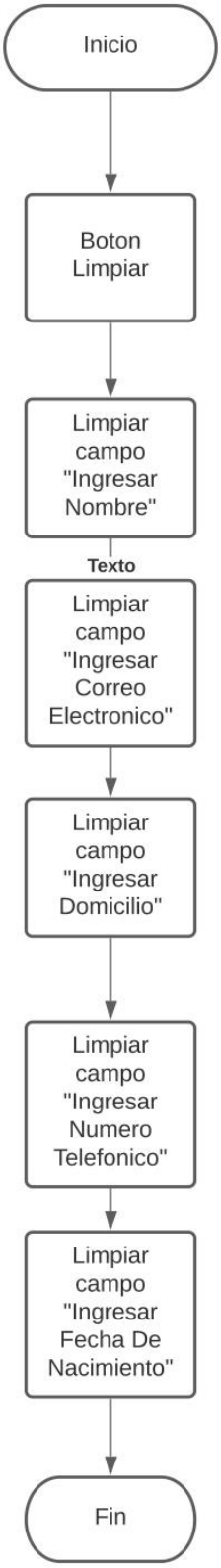
En general, solo puedes usar una variante estática del tiempo de ejecución C++ si tienes solo una biblioteca compartida en la aplicación.

Diagramas

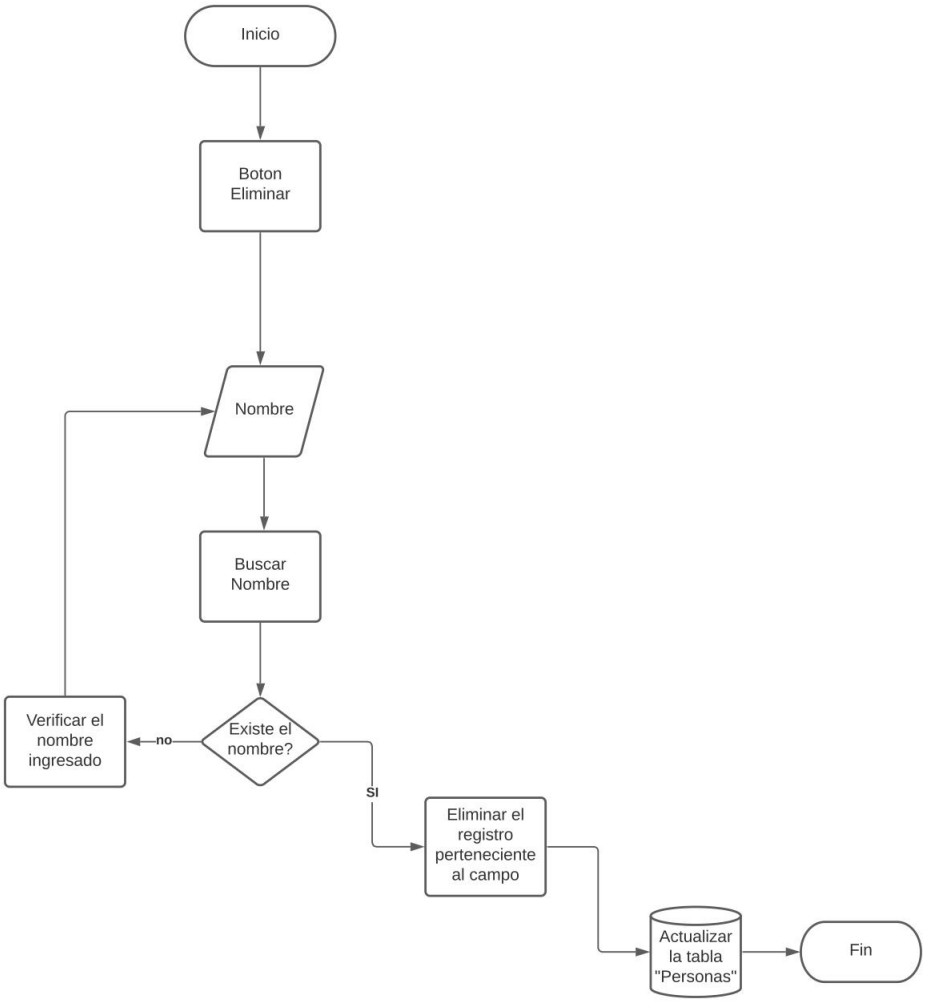
Botón de Mostrar



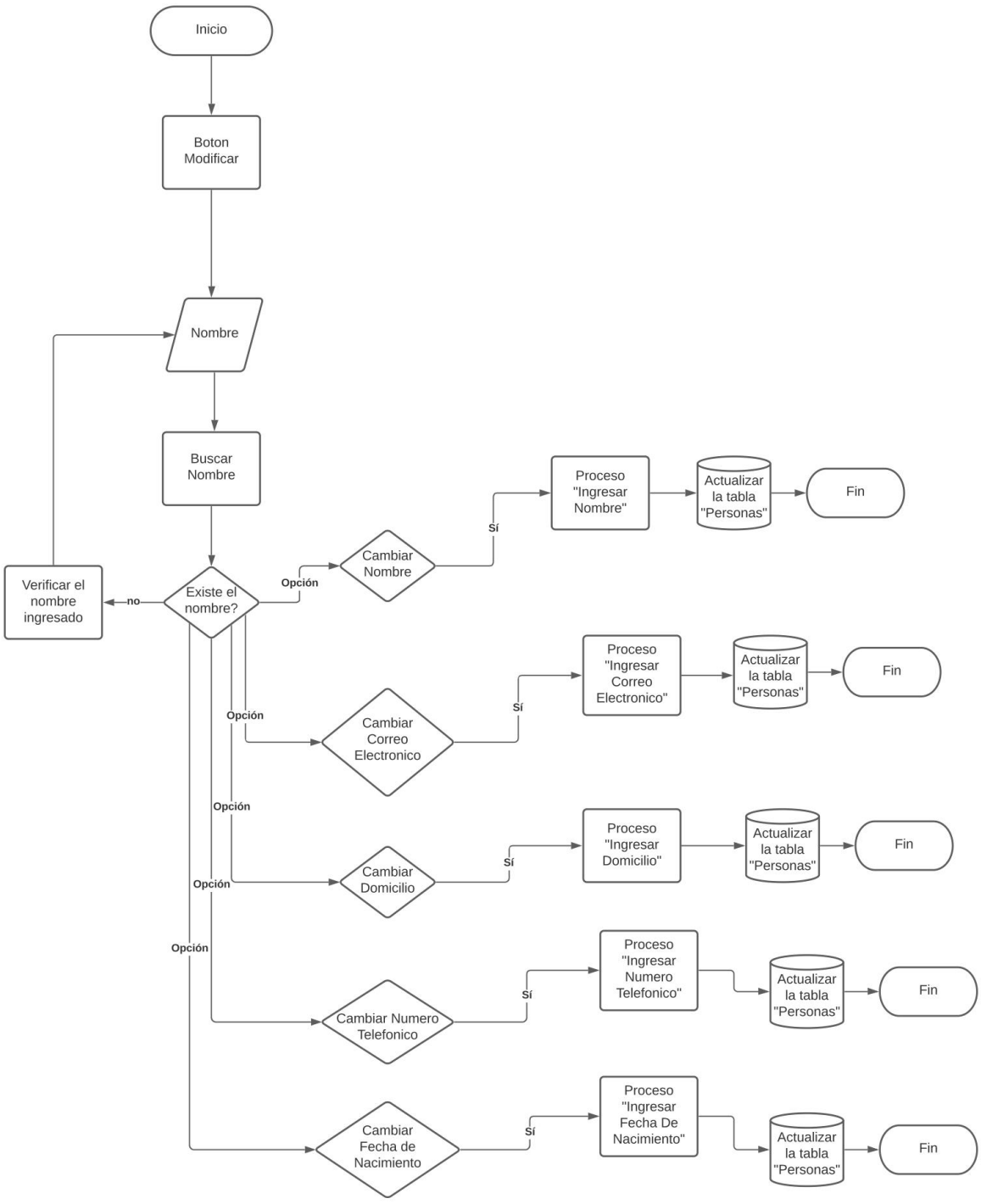
Botón de Registrar



Botón de Limpiar



Botón de Eliminar



Botón de Modificar