# TECNOLÓGICO NACIONAL DE MÉXICO
# INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

## ING. SISTEMAS COMPUTACIONALES

## Lenguajes y autómatas II

## Proyecto 3


**ALUMNO:** Salvador Eliud Carranza de la Rosa

**NO. CONTROL:** 15480101

**CATEDRÁTICO:** Juan Pablo Rosas Baldazo


Cd. Guadalupe, N.L. 16 de Abril del 2018

## Introducción

En la siguiente resumen corresponde al proyecto 3 de la materia Lenguajes y Autómatas II, en el cual se aborda el tema de la optimización,  la optimización consiste en la mejora del código intermedio para que así nos quede un código maquina mejorado y más fácil de ejecutar y con esto se obtiene tanto la optimización temporal como la espacial, se presentan los diferentes tipos de optimización como lo son; las optimizaciones locales, ciclos, globales y de mirilla, se describen que son los costos de ejecución, los criterios para que se pueda mejorar el código así como las herramientas para el análisis del flujo de los datos.

## Capítulo 1: Tipos de optimización

Las optimizaciones pueden realizarse de diferentes formas. Las optimizaciones se realizan en base al alcance ofrecido por el compilador. La optimización va a depender del lenguaje de programación y es directamente proporcional al tiempo de compilación; es decir, entre más optimización mayor tiempo de compilación. La optimización es un proceso que tiene a minimizar o maximizar alguna variable de rendimiento, generalmente tiempo, espacio, procesador, etc.

Dentro de los tipos de optimización se derivan los tipos de optimización local, optimización de ciclo, optimización global y optimización de mirilla.

### Sección 1.1: Locales

La optimización local se realiza sobre módulos del programa. En la mayoría de las ocasiones a través de funciones, métodos, procedimientos, clases, etc. La característica de las optimizaciones locales es que solo se ven reflejados en dichas secciones. La optimización local sirve cuando un bloque de programa o sección es crítico por ejemplo: E/S, la concurrencia, la rapidez y confiabilidad de un conjunto de instrucciones. Como el espacio de soluciones es más pequeño la optimización local es más rápida. Como el espacio de soluciones es más pequeño la optimización local es más rápida.

### Sección 1.2: Ciclos

Los ciclos son una de las partes más esenciales en el rendimiento de un programa dado que realizan acciones repetitivas, y si dichas acciones están mal realizadas, el problema se hace N veces más grandes. La mayoría de las optimizaciones sobre ciclos tratan de encontrar elementos que no deben repetirse en un ciclo.

Ejemplo:

```
while(a == b) {
 int c = a;
 c = 5;
 …;
}
```

En este caso es mejor pasar el Int c =a; fuera del ciclo de ser posible. El problema de la optimización en ciclos y en general radica es que muy difícil saber el uso exacto de algunas instrucciones. Así que no todo código de proceso puede ser optimizado. Otros uso de la optimización pueden ser el mejoramiento de consultas en SQL o en aplicaciones remotas (sockets, E/S, etc.)

### Sección 1.3: Globales

La optimización global se da con respecto a todo el código. Este tipo de optimización es más lenta pero mejora el desempeño general de todo programa. Las optimizaciones globales pueden depender de la arquitectura de la máquina. En algunos casos es mejor mantener variables globales para agilizar los procesos (el proceso de declarar variables y eliminarlas toma su tiempo) pero consume más memoria. Algunas optimizaciones incluyen utilizar como variables registros del CPU, utilizar instrucciones en ensamblador.

### Sección 1.4: De mirilla

La optimización de mirilla trata de estructurar de manera eficiente el flujo del programa, sobre todo en instrucciones de bifurcación como son las decisiones, ciclos y saltos de rutinas. La idea es tener los saltos lo más cerca de las llamadas, siendo el salto lo más pequeño posible.

Instrucciones de bifurcación Interrumpen el flujo normal de un programa, es decir que evitan que se ejecute alguna instrucción del programa y salta a otra parte del programa.

Por ejemplo: el “break”

```
Switch (expresión que estamos evaluando)
{
Case 1: cout << “Hola” ;
Break;
Case 2: cout << “amigos”;
Break;
}
```

## Capítulo 2: Costos

Los costos son el factor más importante a tomar en cuenta a la hora de optimizar ya que en ocasiones la mejora obtenida puede verse no reflejada en el programa final pero si ser perjudicial para el equipo de desarrollo. La optimización de una pequeña mejora tal vez tenga una pequeña ganancia en tiempo o en espacio pero sale muy costosa en tiempo en generarla.

### Sección 2.1: Costo de ejecución (memoria, registros, pilas)

Los costos de ejecución son aquellos que vienen implícitos al ejecutar el programa. En algunos programas se tiene un mínimo para ejecutar el programa, por lo que el espacio y la velocidad de los microprocesadores son elementos que se deben optimizar para tener un mercado potencial más amplio.

Las aplicaciones multimedia como los videojuegos tienen un costo de ejecución alto por lo cual la optimización de su desempeño es crítico, la gran mayoría de las veces requieren de procesadores rápidos (e.g. tarjetas de video) o de mucha memoria. Otro tipo de aplicaciones que deben optimizarse son las aplicaciones para dispositivos móviles.

### Sección 2.2: Criterios para mejorar el código

La mejor manera de optimizar el código es hacer ver a los programadores que optimicen su código desde el inicio, el problema radica en que el costo podría ser muy grande ya que tendría que codificar más y/o hacer su código más legible. Los criterios de optimización siempre están definidos por el compilador.

Muchos de estos criterios pueden modificarse con directivas del compilador desde el código o de manera externa. Este proceso lo realizan algunas herramientas del sistema como los ofuscadores para código móvil y código para dispositivos móviles.

### Sección 2.3: Herramientas para el análisis del flujo de datos

Existen algunas herramientas que permiten el análisis y la correcta optimización del flujo de datos entre las más importantes están:

DEPURADOR
Es una aplicación que permite correr otros programas, permitiendo al usuario ejercer cierto control sobre los mismos a medida que los estos se ejecutan, y examinar el estado del sistema (variables, registros, banderas, etc.) en el momento en que se presente algún problema.

DESAMBLADOR o DESENSAMBLADOR
Es un programa de computadora que traduce el lenguaje de máquina a lenguaje ensamblador, la operación inversa de la que hace el ensamblador.

Un desensamblador se diferencia de un decompilador, en que está dirigido a un lenguaje de alto nivel en vez de al lenguaje ensamblador.

DIAGRAMA DE FLUJO DE DATOS

Es una herramienta de modelización que permite describir, de un sistema, la transformación de entradas en salidas; el DFD también es conocido con el nombre de Modelo de Procesos de Negocios.

DICCIONARIO DE DATOS

El Diccionario de Datos es un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que le permite al usuario y al proyectista del sistema tener una misma comprensión de las entradas, de las salidas, y también de cálculos intermedios.

## Conclusiones

Debido a que los compiladores no lanzan o crean un código tan optimizado este puede ser mejorado por medio de transformaciones a las cuales se le llaman optimizaciones las cuales hacen que el código sea lo mejor posible, el objetivo de la optimización es mejorar el código objeto creado para que este sea más rápido de ejecutar y tenga un mayor rendimiento, lo que hace la optimización es que compensa las ineficiencias del lenguaje fuente, es decir que quita lo que no se necesita o está de más en el código, vimos que hay distintos tipos de optimización y cada una se hace de distintas maneras o en distintas condiciones, uno de los factores más importantes que se debe tomar en cuenta a la hora de hacer la optimización es el de los costos.

## Conceptos

Compilador: es un programa informático que traduce un programa que ha sido escrito en un lenguaje de programación a un lenguaje común, reúne diversos elementos o fragmentos en una misma unidad

Optimización: método para determinar los valores de las variables que intervienen en un proceso o sistema para que el resultado sea el mejor posible

Módulos: es una porción de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos

Funciones: es una sección de un programa que calcula un valor de manera independiente al resto del programa

Métodos: conjunto de instrucciones a las que se les asocia un nombre de modo que si se desea ejecutarlas, sólo basta o referenciarlas a través de dicho nombre en vez de tener que escribirlas

Procedimientos: Porción de código dentro de un programa más grande, que realiza una tarea específica y es relativamente independiente del resto del código

Clases: Es una construcción que permite crear tipos personalizados propios mediante la agrupación de variables de otros tipos, métodos y eventos

Ciclos: es una sentencia que ejecuta repetidas veces un trozo de código, hasta que la condición asignada a dicho bucle deja de cumplirse

Mirilla: proceso de optimización que trata de estructurar de manera eficiente el flujo del programa

Bifurcación: es la creación de un proyecto en una dirección distinta de la principal u oficial tomando el código fuente del proyecto ya existente.

Memoria: puede ser vista como un gran arreglo de bits

Registros: es un tipo de dato estructurado formado por la unión de varios elementos bajo una misma estructura.

Pila: es un tipo de dato estructurado formado por la unión de varios elementos bajo una misma estructura.

Microprocesadores: Procesador de muy pequeñas dimensiones en el que todos los elementos están agrupados en un solo circuito integrado.

## Bibliografía o Referencias

Claudia Dávila. (2014). 3.2.3 HERRAMIENTAS PARA EL ANÁLISIS DEL FLUJO DE DATOS. 14/04/2018, de Prezi Sitio web: https://prezi.com/4dtcp9qnkjbk/323-herramientas-para-el-analisis-del-flujo-de-datos/

Juan Carlos Santiago Hdz. (2016). COSTOS DE EJECUCIÓN. 12/04/2018, de blogspot.mx Sitio web: http://juancarlossant.blogspot.mx/2016/11/costos-de-ejecucion.html

Instituto tecnológico de Piedras Negras. (2013). Unidad III. 11/04/2018, de ITPN Sitio web: http://itpn.mx/recursosisc/7semestre/leguajesyautomatas2/Unidad%20III.pdf

## Reporte

Las optimizaciones se pueden hacer de diferentes formas y se basan al alcance que ofrece el compilador depende del lenguaje de programación y se realiza minimizar el tiempo de ejecución del programa, que use menos espacio y que tenga un mayor rendimiento

Los diferentes tipos de optimización son:

Las optimizaciones locales que son las que se hacen sobre módulos del programa a través de métodos, funciones y procedimientos y solo se ven reflejadas en el módulo que se realizan

Ciclos: estas optimizaciones buscan elementos que no deberían de repetirse en un ciclo ya que si hay un error aquí este se haría más grade por el ciclo ya que se repetiría muchas veces

Las optimizaciones globales se llevan a cabo en todo el código pero son más lentas pero mejoran a todo el programa

Optimizaciones de mirilla este tipo de optimización se lleva a cabo sobre el flujo del programa para hacerlo más eficiente se hace para que el programa no se salte instrucciones a la hora de ejecutarlo

Costos:

Los costos son uno de los factores más importantes que se deben tomar en cuenta al momento de realizar la optimización ya que muchas veces la mejora que se haga puede no reflejarse en el programa final pero si perjudicaría al equipo, es decir, que no habría mejora pero si se consumirían más recursos lo cual no sería factible

Los costos de ejecución son lo que se va a sacrificar para llegar a la eficiencia y eficacia del programa, esto sería la memoria, procesador, espacio etc.

Un ejemplo de esto son los juegos que tienen un costo de ejecución por lo que su optimización de desempeño es muy crítica y por eso necesitan procesadores muy potentes o tarjetas de video de mucha memoria

La mejor forma de la optimización del código es hacer que los programadores optimicen su código desde un principio pero el costo de esto sería que la codificación sea más extensa y el código más legible.