

Clase #15 de 27

Ejemplo de Pragmática – Estilo y buen uso del lenguaje

Agosto 8, Lunes

Agenda para esta clase

- Repaso
 - Niveles del Lenguaje en C
 - GetNextToken de Calculator
- Ejemplo de Pragmática de Stack
- Arreglos y Punteros

Ejemplo de Pragmática

Estilo y buen uso del lenguaje

Análisis Comparativo

```
int Pop(void){  
    int e;  
    e = theElements[ theLevel-1 ];  
    theLevel = theLevel - 1;  
    return e;  
}
```

```
int Pop(void){  
    return theElements[--theLevel];  
}
```

- Más Objetivo
 - Performance
 - Tiempo
 - Accesos
 - Evaluaciones
 - Asignaciones
 - Restas
 - Espacio
 - Memoria
 - Menos Objetivo
 - Estilo correcto
 - Estilo claro
 - Mantenimiento
 - Depuración

- Otra implementación contigua

```
static int theElements[MAX];  
static int* p = theElements;
```

```
int Pop(void){  
    return *--p;  
}
```

- Puntero vs. Índice
 - Equivalencia de expresiones
- Ejemplo de uso de práctico de predecremento
- Analizar
 - Inicialización de p
 - Accesos
 - Evaluaciones
 - Arreglo

Términos de la clase #15

Definir cada término con la bibliografía

- Pragmática
- Equivalencia de expresiones puntero y arreglo
- Aplicación práctica de pre y pos incremento y decremento

Tareas para la próxima clase

1. Dibujar con Dot la máquina de estado asociada a la función GetNextToken ó getop de K&R.



¿Consultas?



Fin de la clase

Clase #16 de 27

Assert & Implementación Enlazada de Stacks

Agosto 22, Lunes

Stack – Interfaz del Módulo

stack.h

```
#ifndef _stackincluded_
#define _stackincluded_
```

```
// Pushes an int
void Push(int);
```

```
// Pops an int
int Pop(void);
```

```
#endif
```

Stack – Implementación Estática

stack.h

```
#ifndef _stackincluded_
#define _stackincluded_

// Pushes an int
void Push(int);

// Pops an int
int Pop(void);

#endif
```

StackStatic.c

```
#include "stack.h"
#include <stdio.h>

const size_t MAX=100;

static int theLevel;
static int theElements[MAX];

void Push(int v){
    if(theLevel < MAX)
        theElements[theLevel++] = v;
    else
        fprintf(stderr,"stack full, "
                "can't push %d\n", v);
}

int Pop(void){
    if(theLevel > 0)
        return theElements[--theLevel];
    else {
        perror("stack empty\n");
        return 0;
    }
}
```

Inicialización de estática
Ocultamiento de información
Static para encapsular
Posincremento ó Incremento
sufijo
Predecremento ó Decremento
prefijo
Aplicación de variables
externas

Uso y Prueba de Stack (int)

StackApplication.c

```
#include "stack.h"
#include <stdlib.h>
#include <stdio.h>

int main(void){
    Push('A');
    Push('B');
    Push('C');
    Push('D');

    putchar(Pop());
    putchar(Pop());
    putchar(Pop());
    putchar(Pop());
}
```

StackTest.c

```
#include "stack.h"
#include <stdlib.h>
#include <assert.h>
```

```
int main(void){
    Muchos ifs anidados con
    condiciones y exit o return
}
```

Compilación de varios archivos fuente
Rol de los headers
los prototipos
Vinculación de archivos obj
Diferencia entre biblioteca y archivo header

Prueba de Stack con Assert

stackTest.c

```
#include "stack.h"
#include <stdlib.h>
#include <assert.h>

int main(void){
    Muchos ifs anidados con
    condiciones y exit o return
}
```

StackTest.c

```
#include "stack.h"
#include <stdlib.h>
#include <cassert.h>

int main(void){
    Push('A');
    Push('B');
    Push('C');
    Push('D');

    assert(Pop()=='D');
    assert(Pop()=='C');
    assert(Pop()=='B');
    assert(Pop()=='A');

    return EXIT_SUCCESS;
}
```

Funcion o macro
Que recibe
Que genera
Como corta
Beneficios a hacerlo
manualmente, numero
y nombre de archivo,
funcion
Test de Unidad
Test Driven Developme

Stack – Implementación Enlazada

StackStatic.c

```
#include "stack.h"
#include <stdio.h>

const size_t MAX=100;

static int theLevel;
static int theElements[MAX];

void Push(int v){
    if(theLevel < MAX)
        theElements[theLevel++] = v;
    else
        fprintf(stderr,"stack full, "
                "can't push %d\n", v);
}

int Pop(void){
    if(theLevel > 0)
        return theElements[--theLevel];
    else {
        perror("stack empty\n");
        return 0.0;
    }
}
```

StackDynamic.c

```
#include "stack.h"
#include <stdlib.h>

struct node{
    int value;
    struct node *next;
};

static struct node *top;

void Push(int v){
    struct node *p;

    if(( p = malloc(sizeof *p) ) == NULL){
        perror("Sin memoria");
        exit(EXIT_FAILURE);
    }
    p->next = top;
    top = p;
    p->value = v;
}

int Pop(void){
    int value = top->value;
    struct node *p = top;

    top = top->next;
    free(p);

    return value;
}
```

struct
Nombre de estructura
Diferencias con C++
Puntero
Función malloc
Operador sizeof
Función exit
Macros simbólicas
EXIT_FAILURE y EXIT_SUCCESS
Función free
Operador ->

Términos de la clase #16

Definir cada término con la bibliografía

- Macro Assert
- Test de Unidad
- Test Driven Development
- Inicialización de variables estáticas
- Ocultamiento de información
- Static para encapsulamiento
- Posincremento ó Incremento sufijo
- Predecremento ó Decremento posfijo
- struct
 - Constante C++
- Puntero
- Función malloc
- Operador sizeof
- Funcion exit
- Macros simbólicas EXIT_FAILURE y EXIT_SUCCESS
- Función free
- Operador ->

Tareas para la próxima clase

1. Leer Lex y Flex.



¿Consultas?



Fin de la clase

Clase #17 de 27

Proceso de Compilación

Septiembre 5, Lunes

Agenda para esta clase

- Proceso de Compilación

El Proceso de Compilación

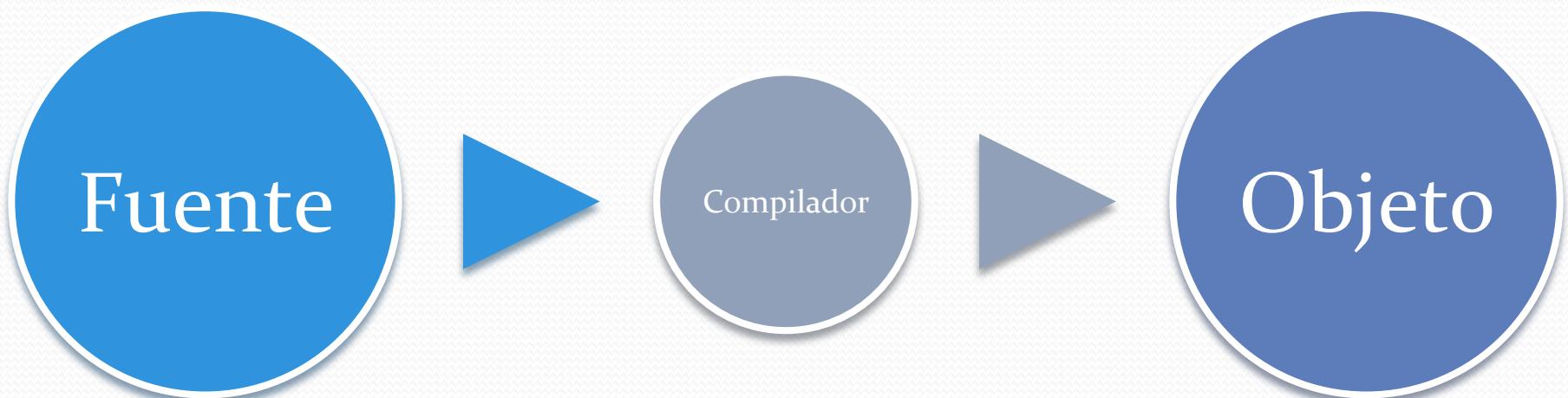
Los Tiempos y Los Ambientes

- Tiempo de
 - Diseño
 - Compilación
 - Ejecución
 - Run-Time
- Ambiente de
 - Traducción
 - Estático
 - Ejecución
 - Dinámico
 - Híbridos
 - Enlace
 - Interpretado
 - Realmente dinámico

Antes y después del Compilador



El Compilador desde afuera – Compilador como Función



Dentro del Compilador



Análisis /
Front End

Síntesis /
Back End

Front End: Análisis por Niveles del Lenguaje

Analizador Léxico / Scanner

Analizador Sintáctico / Parser

Analizador Semántico Estático / Verificador
de restricciones

Analizador Pragmático / Analizador de
código estático: Warnings (e.g., Dead code)

Productos de cada Analizador



Scanner

- Secuencia de caracteres



Parser

- Secuencia de tokens



Semántico
estático

- Árbol de Sintaxis Concreta ó
Árbol de Sintaxis Abstracta

- Árbol de Sintaxis Abstracta Anotado o
Representación Intermedia (IR)

Ejemplos de Árboles Sintácticos Concretos y Abstractos, y Representación Intermedia

- Sentencias
 - If
 - If else
 - For
 - While
- Expresión
 - Suma
 - Subindicación
 - Invocación



Intervalo

20 minutos

Términos de la clase #17

Definir cada término con la bibliografía

- Tiempo de compilación
- Tiempo de ejecución
- Ambiente de traducción
- Ambiente de ejecución
- Antes y después del Compilador
- Compilador como función
- Unidad de traducción
- Compilador como dos etapas
- Front End
- Back End
- Analizadores del Front End
- Scanner ó Analizador Léxico
- Parser ó Análisis Sintáctico
- Analizador Semántico
- Analizador Pragmático
- Árboles Sintácticos Abstractos
- Entrada y salida de cada analizador
- Tokens
- Árbol sintáctico concreto
- Árbol sintáctico abstracto

Tareas para la próxima clase

1. Investigar sobre Expresiones Regulares



¿Consultas?



Fin de la clase

Clase #18 de 27

Expresiones Regulares

Septiembre 12, Lunes

Agenda para esta clase

- Expresiones Regulares

Expresiones Regulares

Definición de los lenguajes de las Categorías Léxicas

- Palabras reservadas
- Puntuación
- Identificadores
- Constantes
 - Entero
 - Flotante
 - Carácter
 - Enumeración
- Literales de cadena.

Ejemplo – Enteros decimales

- $\{1,2,4,3,5,6,7,8,9\} \{0,1,2,3,4,5,6,7,8,9\}^* \{u,U,l,L,\varepsilon\}$
- Necesitamos una forma más compacta
- "Demasiadas llaves y comas".

Definición formal de una ER

- Sintaxis

- $a \in \Sigma$
- E y F representan cualquier ER
- \emptyset es una ER
- ε es una ER
- a es una ER
- $E+F$ es una ER
- EF es una ER
- E^* es una ER
- (E) es una ER

- Semántica

- $L(x)$ significa el LF asociado con x
- $L(\emptyset) = \{\} = \emptyset$
- $L(\varepsilon) = \{\varepsilon\}$
- $L(a) = \{a\}$
- $L(E+F) = L(E) \cup L(F)$
- $L(EF) = L(E) \times L(F)$
- $L(E^*) = L(E)^*$
- $L((E)) = (L(E))$



Intervalo

20 minutos

Términos de la clase #18

Definir cada término con la bibliografía

- Parte I
 - Expresión Regular
 - Operaciones sobre Lenguajes

Tareas para la próxima clase

1. Leer sobre Lex y Flex



¿Consultas?



Fin de la clase

Clase #19 de 27

Regexen & Lex

Septiembre 19, Lunes

Agenda para esta clase

- Expresiones Regulares II
- Regexen
- Lex

Aplicaciones

- Buscar patrones
- Procesar texto
- Herramientas
 - Grep
 - Clases Regex
- Sintaxis Léxica
 - Generación de Analizadores Léxicos.

Operaciones en las ER

- En la definición formal (Precedencia)
 - Unión
 - Concatenación
 - Clausura
 - Paréntesis
- Ejemplo
 - $a+bc^*(d+e)$
 - $a+bc^*d+e$
- Ejercicio: Escribir un BNF para las ER que denote la precedencia. ¿Es ambigua?
- Otras operaciones
 - Clausura Positiva
 - $a^+ = aa^* = a^*a$
 - $(\epsilon+a)^+ = ?$
 - Potencia
 - $a^3 = aaa$
- ¿Otras operaciones de conjunto?
 - Sin complemento
 - Sin intersección.

La ER Universal

- $\Sigma = \{a, b\}$
- $\Sigma^* = \{a, b\}^* = (\{a\} \cup \{b\})^*$
 - Aplicando
 - $L(E+F) = L(E) \cup L(F)$
 - $L(E^*) = L(E)^*$
- $(a + b)^*$

Ejemplos de ER

- Vol 1, página 78,
 - Ejemplo 12
 - $aa+b \stackrel{?}{=} a(a+b)$.
- Ejemplo 16
 - $aa+ab+ba+bb = a(a+b)+b(a+b) = (a+b)(a+b)$
 - ¿Se puede aplicar el operador potencia?
 - ¿Cómo se describe con operaciones de lenguajes?
- Vol 1, página 89, Ejemplo 44
 - $L_1 =$ “todas las palabras que comienzan con a”
 - $a(a+b)^*$
 - $L_2 =$ “todas las palabras que terminan con b”
 - $(a+b)^*b$
 - $L_1 \cap L_2 =$ “todas las palabras que comienzan con a y terminan con b”
 - ¿ER?.



Intervalo

20 minutos

Regexen

AKA:

Regexes

Expresiones Regulares en la Práctica,

Expresiones Regulares,

Expresiones Regulares Extendidas ó

metaER

Ejemplo: ER de las Categorías Léxicas de C

- Constantes enteras hexadecimales
- Constantes reales
- Necesitamos una notación aún más compacta.

Operadores de las Expresiones Regulares Extendidas

- \
 - Escape y especiales
 - [\]\\\]
 - \n
- . (punto)
 - Cualquiera salvo (\n)
 - a.a | (barra vertical)
 - Unión
 - ab|b representa
- Potencias
 - *
 - a*.
- +
 - a+
- { } (llaves)
 - {},
 - a{3}
 - (ab){4}
 - a{1,3}
- Concatenación
 - aa
- ? (interrogación)
 - a?
- () (paréntesis)
 - Ejemplo: ((ab)? | b)+
- Enumeraciones
 - [] (corchetes)
 - Enumeración
 - [abx]
 - - (guión)
 - [a-d]
 - [o-9a-z]

Ejemplos de Expresiones Regulares Extendidas

- Vol 1, Página 89, Ejemplo 44
 - $[0-9]^* \cdot [0-9]^+$
- Ejemplo 50
 - $(\cdot + | \cdot -)^? [0-9]^+$
- Ejemplo 51
 - (a) La metaER $[ab][cd]$ equivale a $(a+b)(c+d) = ac + ad + bc + bd.$
 - (b) La metaER $[ab]c$ equivale a $(a+b)c = ac + bc$
 - (c) $[ab][ba] .$

Ejercicios

- Vol 1, Página 95, Ejercicio 55
 - Escriba una metaER que sea equivalente a la ER $e+(a+b)^34$
- Ejercicio 59
 - Dado el alfabeto de los dígitos decimales, obtenga una metaER que represente al LR “Todas las palabras que terminan con 22 o con 47 y tienen longitud mayor o igual que 3”.

Notaciones y Formalidades

- Teoría
 - Expresiones Regulares
 - Lenguajes Formales
- Práctica
 - SRE
 - BRE
 - ERE
 - Backreference y otras extensiones para lenguajes no regulares



Lex

Lex

- def.l (reglas) [**Lex**] lex.yy.c
- lex.yy.c [**Compilador c**] lex.yy.exe
- Input [**yy.lex.exe**] output
- Ejemplo de copy
- Usar el main por defecto

Ejemplo mínimo

```
%option main  
%%
```

Copiar entrada a salida; 2da versión

[K&R1988] 1.5

```
#include <stdio.h>          %option main
int main(void){              %%
    int c;
    while( (c=getchar()) != EOF )
        putchar(c);
    return 0;
}
```

//

Secciones de un programa lex

- General
 - ... definiciones...
 - %%
 - ... reglas...
 - %%
 - ... subrutinas...
- Secciones opcionales
 - Segundo %%
 - Regla por defecto
- Mínimo Lex
 - %%
- Mínimo Flex
 - %option main
 - %%

Patrones Lex

[MUCH2012] vic5p92

- . any character but newline
- x|y an x or a y
- [xy] the character x or y
- [x- z] the characters x, y or z
- x{n} n occurrences of x
- x{m,n}m through n occurrences of x
- x? an optional x
- x* 0,1,2, ... instances of x
- x+ 1,2,3, ... instances of x
- (x) an x

Otros

- x the character "x"
- "x"an "x", even if x is an operator
- \x an "x", even if x is an operator
- [^x] any character but x
- ^x an x at the beginning of a line
- x\$ an x at the end of a line
- <y>x an x when in start condition y
- x/y an x but only if followed by y
- {xx} the translation of xx from the definitions section

Conteo de caracteres; 2da versión

[K&R1988] 1.5.2

```
#include <stdio.h>
int main(void){
    double nc;
    for(
        nc=0 ;
        getchar()!= EOF ;
        ++nc
    )
        ;
    printf("%.0f\n", nc);
    return 0;
}
%option noyywrap
%{
double nc;
%%
.\n    ++nc;
%%
int main(void){
    yylex();
    printf("%.0f\n", nc);
    return 0;
}
```

Conteo de líneas [K&R1988] 1.5.3

```
#include <stdio.h>
int main(void){
    int c, nl;
    nl = 0;
    while(
        (c=getchar())!=EOF
    )
        if (c == '\n')
            ++nl;
    printf("%d\n", nl);
    return 0;
}
%option noyywrap
%{
double nl;
%%
\nl  ++nl;
.   ;
%%
int main(void){
    yylex();
    printf("%.0f\n", nl);
    return 0;
}
```

Conteo de dígitos, blancos y otros

[K&R1988] 1.6

```
#include <stdio.h>

int main(void){
    int c, i, nwhite, nother;
    int ndigit[10];

    nwhite = nother = 0;
    for ( i = 0; i < 10; ++i )
        ndigit[i] = 0;

    while ( (c = getchar()) != EOF )
        if ( c >= '0' && c <= '9' )
            ++ndigit[c - '0'];
        else if(c==' '||c=='\n'||c=='\t')
            ++nwhite;
        else
            ++nother;

    printf("digits =");
    for (i = 0; i < 10; ++i)
        printf(" %d", ndigit[i]);
    printf(", whitespace=%d, other=%d\n",
           nwhite, nother);
    return 0;
}
```

```
%option noyywrap
%{
int nwhite, nother, ndigit[10];
%}
%%
[0-9]    ++ndigit[ yytext[0] -
'0' ];
[ \n\t]   ++nwhite;
.         ++nother;
%%
int main(void){
    int i;
    yylex();
    printf("digits =");
    for(i = 0; i < 10; ++i)
        printf(" %d", ndigit[i]);
    printf(", whitespace=%d,
other=%d\n", nwhite, nother);
    return 0;
}
```

Longitud Promedio de Líneas

```
//  
int main(void) {  
    int nl, nc;  
  
    for(nl=0,nc=0;(c=getchar())!=EOF;)int nl, nc;  
        if(c == '\n')  
            ++nl;  
        else  
            ++nc;  
  
    if( !feof(stdin) )  
        perror("Error!");  
  
    printf("%.1f\n",nl?nc/(float)nl:0); int i;  
  
    return EXIT_SUCCESS;  
}  
  
%option noyywrap  
%{  
    \n    ++nl;  
    .    ++nc;  
%%  
int main(void){  
    yylex();  
    printf("%.1f\n",nl?nc/(float)  
0);  
    return 0;  
}
```

Conteo de Palabras [K&R1988]

1.5.4

```
#include <stdio.h>
#define IN 1 /* inside a word */
#define OUT 0 /* outside a word */

int main(void){
    int c, nl, nw, nc, state;
    state = OUT; nl = nw = nc = 0;

    while ((c = getchar()) != EOF) {
        ++nc;
        if (c == '\n')
            ++nl;
        if (c==' ' || c=='\n' || c=='\t')
            state = OUT;
        else if (state == OUT) {
            state = IN;
            ++nw;
        }
    }

    printf("%d %d %d\n", nl, nw, nc);
    return 0;
}
```

```
%option noyywrap
%{
int nl, nc, nw;
%}
WORD  [^ \t\n]+
%%%
{WORD} {++nw;nc+=yylen;}
\n      {++nl;++nc;}
[ \t]   ++nc;
%%%
int main(void){
    yylex();
    printf("%d %d %d
\n",nl,nw,nc);
    return 0;
}
```

Contar Vol 1, Página 96-7, Ejemplo

53

```
%{
/* Reconoce números enteros y palabras alfabéticas;
además, cuenta las ocurrencias de ambos */
#include <stdio.h>
int nros = 0, pals = 0;
}%
%[
[0-9]+ { nros++; }
[a-zA-Z]+ { pals++; }
.|\\n ;
%}
int main(void) {
    yylex();
    printf("Se reconocieron:\\n");
    printf("%d Numeros y\\n", nros);
    printf("%d Palabras.\\n\\n", pals);
    return 0;
}
```

Lexema Vol 1, Página 95-6, Ejemplo

52

```
%{  
/* Detecta e imprime los números enteros */  
#include <stdio.h>  
}  
%%  
[0-9]+ { printf("%s\n", yytext); }  
.|\n;  
%%  
int main(void) {  
    yylex();  
    return 0;  
}
```

Otra versión, con definiciones

```
%{  
/* Detecta e imprime los números enteros */  
#include <stdio.h>  
%}  
D      [0-9]  
%%  
{D}+    { printf("%s\n", yytext); }  
.|\n ;  
%%  
int main(void) {  
    yylex();  
    return 0;  
}
```

Términos de la clase #19

Definir cada término con la bibliografía

- Parte I
 - Precedencia de operadores de expresiones regulares
 - Clausura positiva
 - Potencia
 - Sin complemento
 - Sin intersección
 - ER Universal
- Parte II
 - Regexen
 - metaER
 - Clase de lenguajes de las Regexen
- Parte III
 - Lex
 - Especificación Lex
 - Definición, Entrada de Lex
 - Salida de Lex
 - Patrón dado por Regex
 - Acción dada por sentencia C
 - Regla por defecto
 - Orden de evaluación de reglas
 - Sección #2: Reglas
 - Sección #1: Definiciones de archivo entrada para Lex
 - Sección #3: funciones

Tareas para la próxima clase

1. Prácticar Lex



¿Consultas?



Fin de la clase

Clase #20 de 27

Lenguajes Formales & Autómata de Pila

Septiembre 26, Lunes

Agenda para esta clase

- Conceptos fundamentales de Lenguajes Formales
- Definición de Lenguajes Formales
- Lenguajes No Regulares
- Introducción a Autómatas de Pila



Símbolo, Alfabeto & Cadena

Símbolo o Carácter

- Es Indivisible
- Es cualquier elemento que se defina como símbolo
- Pertenece a un Alfabeto (Σ)
- Ejemplo [Mvíci]3: El alfabeto $\Sigma = \{0, 1\}$ proporciona los caracteres para construir los números binarios
- Ejemplo: El $\Sigma = \{\text{if, for}\}$ tiene dos caracteres
- Ejercicio [Mvíci]1: Escriba el Σ para construir el conjunto de los números enteros con signo en base 10
- Operaciones
 - Concatenar dos caracteres
 - a con $b = ab$
 - Potencia7
 - $a^7 = aaaaaaa$
 - $\text{for}^7 = ?$
 - ¿Cerradas?

Alfabeto Σ

- Es un Conjunto, cuyos elementos son símbolos o caracteres
- Restricciones básicas
 - Finito
 - No vacío
- Ejemplos
 - $\{\} = \emptyset \Rightarrow$ no es alfabeto, por ser vacío
 - Naturales \Rightarrow no es alfabeto, por ser infinito
 - ASCII
 - Dígitos $= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
 - Meses $= \{\text{Lunes, Martes, Miércoles, Jueves, Viernes, Sábado}\}$
- Restricción avanzada
 - Sus elementos no pueden formarse por yuxtaposición de otros elementos
- Ejemplos
 - $\{0, 1, 1001\}$
 - $\{\text{if, for}\}$
 - $\{\text{a, b, c, ch, d, e, f, g, h, i, j, k, l, ll, m, n, ñ, o, p, q, r, rr, s, t, u, v, w, x, y, z}\}$

Caracteres y Alfabetos – Implementación en ANSI C

- Alfabeto tipo de dato char
- Carácter valor y variable
 - A
 - Literal 'A'
 - Literal 65
 - $c \in \Sigma$
 - char c;
- Concatenar dos caracteres
 - ¿Uso?
 - void DosCaracteres(int, int, char *);
- Potencia 7
 - ¿Uso?
 - void SieteVeces(int, char *);

Cadena o String

- Secuencia finita
 - $\Sigma=\{a, b\}$
 - aab
 - Pares ordenados y Producto Cartesiano
 - ((a, a), b)
 - N-Tupla
 - (a, a, b)
 - Ejercicio 3: Dado el $\Sigma=\{\text{if}, \text{for}\}$, construya una cadena que tenga cuatro caracteres
- Algunas Operaciones
 - Concatenación
 - Longitud
 - Potenciación
 - Inversa
 - Instancias de un carácter
 - Contiene tal carácter
 - ¿Cuáles son cerradas?

Cadenas especiales

- Secuencia vacía
 - ()
 -
- Símbolos
 - Épsilon ϵ ó
 - Lambda λ
- ¿A qué alfabeto pertenece?
 - (Evitar la respuesta obvia)
- En ANSI C
 - """
- Un solo carácter
 - a (carácter)
 - En ANSI C
 - 'a'
 - a (string)
 - En ANSI C
 - "a"

Tipo de dato de una cadena

- Valores, Variables, y Tipo de Dato
 - ¿ Tipo de Dato Matemático ?
 - a
 - aa
 - aaaaaaaaa
 - ϵ
 - ¿ Tipo de Dato C?
 - "a"
 - "aa"
 - "aaaaaaaa"
 - ""
- ¿Tipo de dato de las cadenas en el Lenguaje Pascal?
- ¿Tipo de dato de las cadenas en el Lenguaje C?

Tipo de dato de cualquier cadena

$$\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots \cup \Sigma^{255}$$

$$\bigcup_{i=0}^n \Sigma^i$$

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$$

Lenguajes Formales

Lenguajes Naturales & Formales

- Lenguajes Naturales
 - Para comunicarse entre personas
 - Evolución continua
 - Reglas gramaticales y sintaxis surgen después
 - El pragmatismo se puede obtener aunque la sintaxis o la semántica no sea la correcta
 - Preciso
- Lenguajes Formales
 - Para comunicarse entre sistemas o para formalizar conceptos
 - Evolución discreta
 - Reglas gramaticales y sintaxis surgen primero
 - El pragmatismo se obtiene solo si la sintaxis y la semántica son correctas
 - Ambiguo

Lenguajes Formales

- Conjunto de cadenas
- Sus elementos se llaman palabras
- Definición
 - Por extensión
 - Por comprensión

Ejemplo 27

Sea el siguiente Lenguaje Formal descripto por EXTENSIÓN: $L = \{101, 1001, 10001, 100001\}$. Utilizando el operador "supraíndice", este lenguaje puede ser descripto por COMPRENSIÓN, en forma más compacta, así: $L = \{10^n1 / 1 \leq n \leq 4\}$. Entonces, la cadena 1001 ("uno-cero-cero-uno") es una palabra del lenguaje L , mientras que 1100 ("uno-uno-cero-cero") es una cadena construida con caracteres del mismo alfabeto pero no es una palabra de L .

- Diagramas sintácticos
- Expresiones regulares
- Máquinas de estado como Autómatas Finitos
- Ejemplo [Mv1c1] 27.

Ejemplo 31 [Mv1c1]

- $L = \{(abc)^n / 0 \leq n \leq 3\}$
 $L = \{\epsilon, abc, abcabc, abcabca\}.$
- ¿Alfabeto?
- ¿ $|abc| = 3$?
- ¿Concatenación o potencia cerrada?
 - La palabra vacía es un miembro L
 - La concatenación de las palabras abc y abcabc produce otra palabra de este lenguaje: abcabca.
 - En cambio, la concatenación de la palabra abcabc consigo misma produce la cadena abcabcaabc, que no es una palabra de este lenguaje.
 - La potencia $(abc)^2$ es una palabra del lenguaje
- Definir L por medio de LN
 - La cadena abc hasta tres veces o ninguna vez

Ejemplo 32 [Mv1c1]

- $L = \{a^{2n+1} / 0 \leq n \leq 200\}$
¿Concatenación cerrada?
- Ejercicio 15: Definir L por medio de LN

Cardinalidad de un Lenguaje Formal

- Ejemplos
 - $L_1 = \{a, ab, aab\}$
 - $L_2 = \{\} = \emptyset$
 - $L_3 = \{\varepsilon\}$

Sublenguajes

- Sublenguajes de un Lenguaje
 - \emptyset es sublenguaje de todo lenguaje
- ¿Cómo se expresa "Todos los Sublenguajes de un Lenguaje"?
 - Conjunto potencia: $P(L)=2^L$
- Ejercicio. Completar:
 - $L = \{a, b\}$
 - $2^L = \{$
 - $|2^L| = |2|^{|L|} =$

Lenguajes Formales – El Lenguaje Universal sobre un Alfabeto – Definición formal de Σ^*

- $\Sigma = \{a, b\}$ Alfabeto
- $\Sigma^0 = \{\epsilon\}$
Lenguaje de palabras de longitud cero
 - $\Sigma^0 \neq \emptyset$
 - $\emptyset = \{\}$
- $\Sigma = \Sigma^1 = \{a, b\}$
Lenguaje de palabras de longitud uno
- $\Sigma \times \Sigma =$
 $\Sigma^2 = \{aa, bb, ab, ba\}$
Lenguaje de palabras de longitud dos

- $\Sigma \times \Sigma \times \Sigma =$
 $\Sigma^2 \times \Sigma =$
 $\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$
Lenguaje de palabras de longitud tres
- $\Sigma^* =$
 $\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots =$

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$$

Lenguajes Formales – Dado un Σ

¿Cuántos lenguajes?

- [Mv1c1] Ejercicio 10:
Dado el LF = {Argentina, Holanda, Brasil}
Indique el Σ (alfabeto) mínimo
- Dado el Σ anterior, indique cuántos lenguajes de cada cardinalidad existen
 - 0
 - 1
 - 2
 - n
 - Infinita



Intervalo

20 minutos

Introducción a Autómata Pushdown

Máquinas de Estado con Mayor Poder de Cómputo

Ejemplo – Balanceo de Paréntesis, Corchetes, y Llaves

- $(())$
 - Una AF no es suficiente
 - Agregar contador
 - $\{ \} \}$
 - Dos contadores
 - No es suficiente
 - $\{ () \}$
 - Pila
 - Gramática del LF
 - $L=\{a^nb^n/n>0\}$
 - $S \rightarrow ab$
 - $S \rightarrow aSb$
-
- Solución
 - Autómata de Pila
 - Autómata Finitio de Pila (AFP)
 - Pushdown Automata (PDA)
 - Versión determinística
 - AFPD
 - Versión no-determinística, más poderosa
 - AFPND
 - Versión que reconoce por
 - Estados finales
 - Pila vacía
 - Equivalentes
 - Jerarquía de Chomsky
 - Transición
 - Diagrama de Transición
 - Pop de carácter
 - Push de string
 - Centinela para marcar pila vacía

Términos de la clase #20

Definir cada término con la bibliografía

- Parte I
 - Símbolo
 - Alfabeto
 - Cadena
- Parte II
 - Lenguaje Universal
 - Clasrusua de Kleene
- Parte III
 - Autómata de Pila
 - Diagrama de estados para autómata de Pila

Tareas para la próxima clase

1. Leer de [MUCH2012] v2c5 Autómatas de Pila.



¿Consultas?



Fin de la clase