Clase #06 de 27
Entrada y Salida de a Caracteres

Abril 17, Lunes

Repaso Clase Anterior

- Glosario
- Tareas

Agenda para esta clase

- Flujos y copia
- Conteo de caracteres
- Flujos de Texto y Líneas
- Selección de caracteres

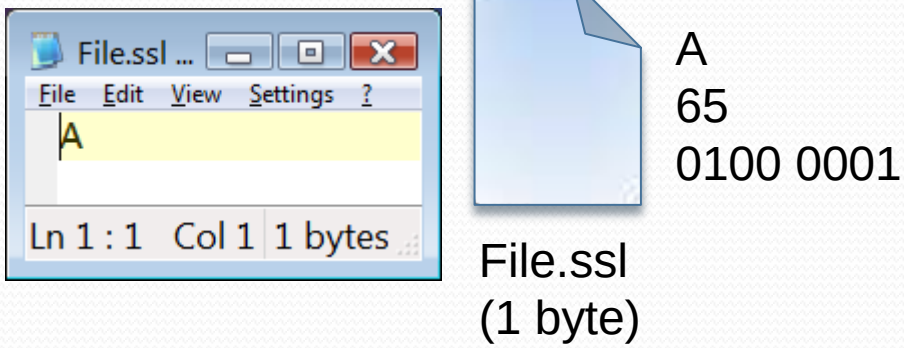
Entrada y Salida con Flujos de Texto Standard

K&R 1.5 Entrada y Salida de a Caracteres

Hasta 1.5.3

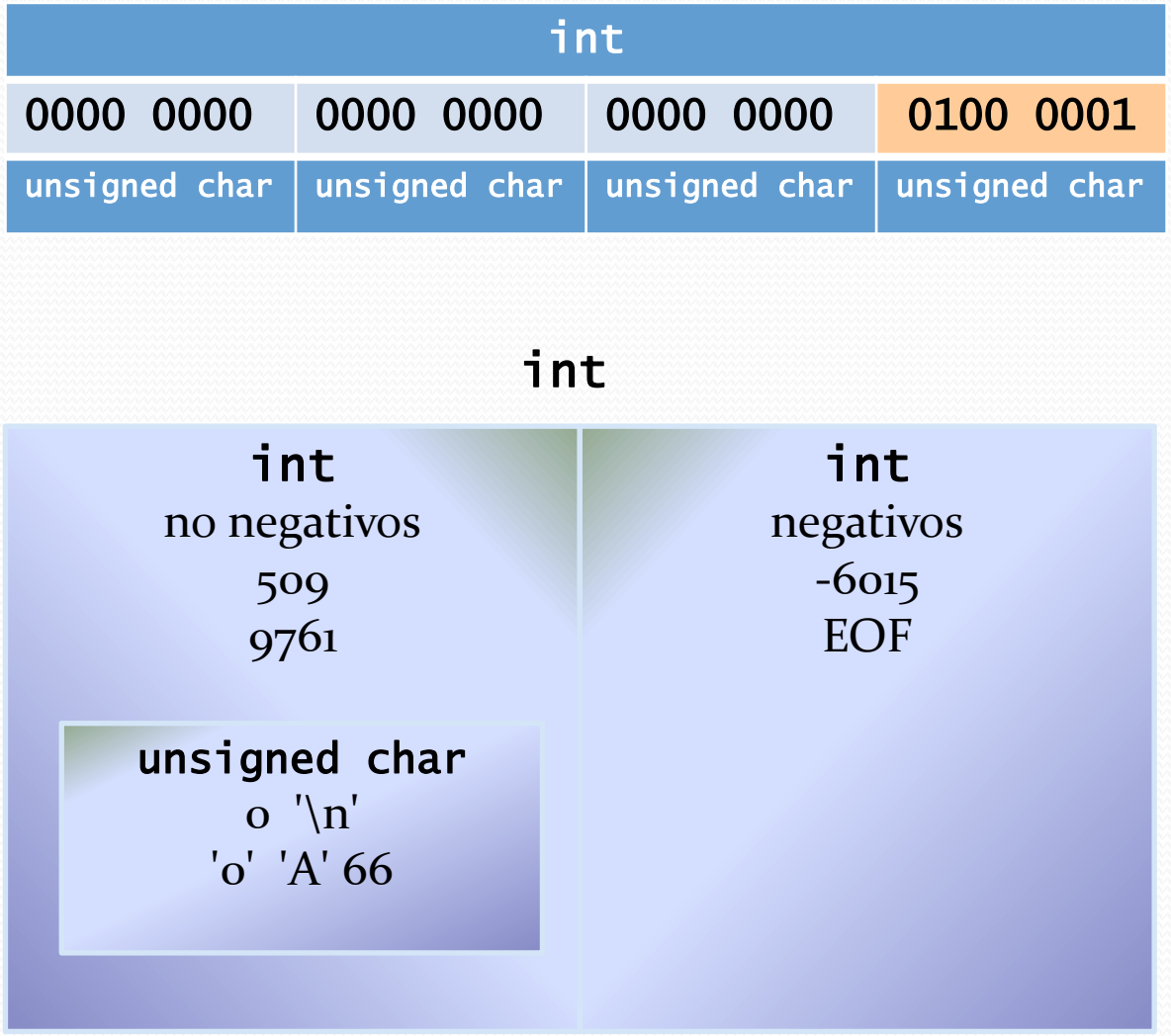
Flujos (Streams) y Archivos

- Proceso como sistema abierto
- Flujo de datos: Origen o destino de datos
  - Secuencia de Datos
  - Origen
    - Archivo
    - Otro Procesos
    - Teclado u otros dispositivos
  - Destino
    - Archivo
    - Otro Procesos
    - Pantalla u otros dispositivo
- Diferencia entre flujo y archivo
  - Durante la lectura
    - ¿Cambia el archivo?
    - ¿Cambia el flujo?
  - Durante la escritura
    - ¿Cambia el archivo?
    - ¿Cambia el flujo?
- ¿Existe diferencia entre un "Archivo Binario" y un "Archivo de Texto"?



Primitivas para operar con Flujos

- Un flujo es una secuencia de bytes
  - Relación entre char y byte
- Primitiva putchar
  - Recibe un char no signado dentro de un int
  - Retorna EOF si no pudo enviar el dato
- Primitiva getchar
  - Retorna un char no signado dentro de un int, ó el int EOF
- EOF es una Señal
  - Valor abstraído
    - Es un int negativo
  - Semántica
    - Fin por finalización
    - Fin por error.



Algoritmo de Lectura para Recorrer un Flujo

- Problema “Implementar la función identidad”: Sale lo que entra, lo que entra sale
- ¿Cuándo finaliza de procesar?
- ¿Cómo se programa en Pascal?
  - ¿Cuál es el diagrama N-S?
- ¿Hay que hacer una lectura previa?
- ¿Cómo se programa en C++?
  - ¿Cuál es diagrama N-S?
- ¿Cómo se programa en C?
  - ¿Cuál es diagrama N-S?.

Copiar entrada a salida; 1era versión

```
Leer un carácter
Mientras el carácter no
sea el indicador de fin
  Mostrar ese carácter
  Leer otro carácter
```

```
#include <stdio.h>

// copy input to output; 1st version
int main(void){
  int c;

  c = getchar();
  while (c != EOF) {
    putchar(c);
    c = getchar();
  }
}
```

Copiar entrada a salida; 2da versión

```
#include <stdio.h>

// copy input to output; 2nd version
int main(void){
  for (int c; (c = getchar()) != EOF;)
    putchar(c);
}
```

```
Mientras(haya caracteres)
    Enviar carácter
```

```
/* C89: */
int main(void){
  int c;

  while ( (c = getchar()) != EOF)
    putchar(c);
}
```

- "*Idiom*", Expresión idiomática, frase hecha
- Precedencia
- Expresiones equivalentes
  - `c = getchar() != EOF`
  - `c = (getchar() != EOF)`

Ejercicios

- 1-6. Verifique que la expresión getchar() != EOF es cero o uno
- 1-7. Escriba un programa que imprima el valor de EOF.

Conteo de Caracteres

K&R 1.5.2

Conteo de caracteres; 1era versión

```
#include <stdio.h>

// count characters in input; 1st version
int main(void){
  long nc;

  nc = 0;
  while (getchar() != EOF)
    ++nc;
  printf("%ld\n", nc);
}
```

Conteo de caracteres; 2da versión

```
#include <stdio.h>

// count characters in input; 2nd version
int main(void){
  double nc;

  for (nc = 0 ; getchar() != EOF; ++nc)
    ;
  printf("%.0f\n", nc);
}
```

Flujos de Texto & Líneas

K&R 1.5.3

Flujos de Texto

- Definición de Flujo de texto
  - Secuencia de líneas
    - Secuencia de caracteres finalizada por el carácter nueva línea ('\n')
- Flujos de texto estándar
  - Entrada: stdin
  - Salida: stdout
  - Error: stderr
- Redirección
  - hola.exe > salida.txt
  - ordenar.exe < in.txt > out.txt

- La definición del **modelo** de flujo de texto es **única** y conocida
- Pero **cada entorno** de ejecución tiene su **propia representación**
- La **biblioteca abstrae** los detalles de implementación y **presenta al programador** los flujos de texto de una **única forma**, la del modelo.

Abstracción por medio del modelo

- Diferentes representaciones de líneas de texto
  - Mac OS 9
    - CR 13
  - Windows
    - CR+LF 13 10
  - Unix y derivados (Mac OS X)
    - LF 10
  - Muchas representaciones más, por ejemplo
    - Longitud
    - Fija con Espacios
- Modelo
  - Abstracción de la representación
  - Responsabilidad.

Ejemplo de aplicación de la abstracción

- Origen
  - Conceptual
    - ABC
    - DE
  - Modelo
    - A B C \n D E \n
    - 65 66 67 10 68 69 10
- Lectura

```
while( (c=getchar()) != EOF )
  printf("%d ", c);
```

- Diferentes implementaciones
  - Mac Os 9
    - Origen65 66 67 13 68 69 13
    - Salida 65 66 67 10 68 69 10
  - Windows
    - Origen65 66 67 13 10 68 69 13 10
    - Salida 65 66 67 10 68 69 10
  - Unix, Mac OS X
    - Origen65 66 67 10 68 69 10
    - Salida 65 66 67 10 68 69 10
  - Hipothetic OS (Longitud)
    - Origen3 65 66 67 4 68 69
    - Salida 65 66 67 10 68 69 10

Diferencia entre Flujos de Texto y Flujos Binarios

Escritura por Flujo de Texto

- Texto
  - ABC
  - DE
- Representación según modelo
  - A B C \n D E \n
- Destino: Archivo "buffer.ssl"
- Tareas
  - Escribir un programa que imprima las dos líneas
  - Determinar la representación en bytes del archivo para cada entorno
    - Entorno Windows
    - Entorno Mac OS 9
    - Entorno Hypothetic OS
      - Longitud representada con unsigned int de 32 bits
    - Entorno Fixed Maximum Width
      - Determinar delimitador, tamaño y relleno

Leer desde de Flujo de Texto o de Flujo Binario

- Código fuente
  - ¿Cómo mostrar el entero asociado a cada carácter?
- Origen: Archivo "buffer.ssl"
- Lectura a través de flujo de texto
- Lectura a través de flujo binario
- Conclusión
  - ¿Cuál es la diferencia?
    - Conversión
- Tareas
  - Escribir programa para mostrar valor de cada byte
  - Determinar salida de ese programa para cada entorno, para flujo de texto y para flujo binario
    - Entorno Windows
    - Entorno Mac OS 9
    - Entorno Hypothetic OS
    - Entorno Fixed Maximum Width

Demostración con ContarLineas

```
#include <stdio.h>

// count lines in input
int main(void){
   int c, nl;

   for (nl = 0; (c = getchar()) != EOF;)
      if (c == '\n')
         ++nl;
   printf("%d\n", nl);
}
```

- Lietarl Carácter o Constante Carácter
- Valor de un Literal Carácter
- Tipo de un Literal Carácter
- Buffer
  - Límite
  - Comienzo del procesamiento
- Señal de fin de ingreso
- Redirección
  - >
  - <
  - Pipes |
- Ejemplos
  1. `cat lc.c`
  2. `make lc`
  3. `find . -name "lc*"`
  4. `ls lc.*`
  5. `./lc`
  6. `./lc < lc.c`
  7. `./lc < lc.c > out.txt`
  8. `cat out.txt`
  9. `./lc < lc.c | ./lc`

Selección de Caracteres

Ejercicios

- 1-8. Escriba un programa que cuente blancos, tabs y nuevalineas
- 1-9. Escriba un programa que copie su entrada en su salida, reemplazando cada secuencia de uno o más blancos por un solo blanco
- 1-10. Escriba un programa que copie su entrada en su salida, reemplazando cada tab por \t, cada retroceso por \b, y cada barra invertida por \\. Esto hace que los tabs y los retrocesos sean visibles sin ambigüedades.

1-8. Contar blancos, tabs y nuevalineas: Dos Resoluciones

```
int main(void){
    unsigned ns, nt, nl;

    ns = nt = nl = 0;
    for(int c; (c = getchar()) != EOF;)
        if(' ' == c)
            ++ns;
        else if('\t' == c)
            ++nt;
        else if ('\n' == c)
            ++nl;
    printf("ns = %d, nt = %d, nl =
%d\n", ns, nt, nl);
}
```

```
int main(void){
    unsigned ns, nt, nl;

    ns = nt = nl = 0;
    for(int c; (c = getchar()) != EOF;)
        switch(c){
            case ' ':
                ++ns;
                break;
            case '\t':
                ++nt;
                break;
            case '\n':
                ++nl;
        }
        printf("ns = %d, nt = %d, nl =
%d\n", ns, nt, nl);
}
```

Términos de la clase #6

Definir cada término con la bibliografía

- Flujos y copia
  - Stream (Flujo de datos)
  - Archivo versus Flujo
  - stdin
  - stdout
  - "Diferencia" entre Archivo Binario y Archivo de Texto
  - Primitivas para operar con flujos
  - putchar
  - getchar
  - EOF
  - “Idiom”, Frase hecha, Expresión idiomática
  - Idiom para recorrer flujo de a caracteres
  - Precedencia (ó Prioridad) de operadores
- Conteo de caracteres
  - Tipo long
  - Tipo double
  - Formato long
  - Formato double
- Flujos de Texto y Líneas
  - Flujo de texto
  - Línea
  - Nueva Línea
  - Flujos de texto estándar
  - stderr
  - Redirección de la entrada y de la salida
  - Abstracción de la Representaciones de flujos de texto
  - Diferencia entre Flujo Binario y Flujo de Texto
  - Flujo binario
  - Encadenamiento o entubamiento (Pipe) de comandos, la salida de un proceso a la entra de otro
  - make sin makefile
  - Carácter literal o constante carácter
  - Valor y tipo de un literal carácter
  - Entrada con bufferr
- Selección de Catacteres
  - Introducción switch.

Tareas para la próxima clase

1. Análisis comparativo de if versus switch
2. Estudiar v2c1 Introducción a Autómatas Finitos y Aplicaciones de [MUCH2012]
3. Estudiar Comentarios y Categorías Léxicas de
    1. [MUCH2012]
    2. [K&R1988].

¿Consultas?

Fin de la clase

Clase #07 de 27

Sentencias de Selección, Árboles de Expresión, & Máquinas de Estado I

Abril 24, Lunes

Repaso Clase Anterior

- Glosario
- Tareas

Agenda para esta clase

- Sentencias de Selección: If versus Switch
- Árboles de Expresión
- Máquinas de Estado I

Sentencias de Selección

If versus Switch

Sentencias de Selección, Etiquetada y de Salto

sentencia-de-selección:

`if (` *expresión* **`)`** *sentencia*

`if (` *expresión* **`)`** *sentencia* **`else`** *sentencia*

`switch (` *expresión* **`)`** *sentencia*

sentencia-de-salto

`continue ;`

`break ;`

`return` *expresión?* **`;`**

`goto` *identificador* **`;`**

sentencia-etiquetada

`case` *expresión-constante* **`:`** *sentencia*

`default :` *sentencia*

identificador **`:`** *sentencia*

If versus Switch: Pocos casos

```
if(' ' == c)
    ++ns;
  else if('\t' == c)
    ++nt;
  else if ('\n' == c)
    ++nl;
```

```
        cmp     DWORD PTR [rbp-16], 32
        jne     .L3
        add     DWORD PTR [rbp-4], 1
        jmp     .L6
.L3:
        cmp     DWORD PTR [rbp-16], 9
        jne     .L5
        add     DWORD PTR [rbp-8], 1
        jmp     .L6
.L5:
        cmp     DWORD PTR [rbp-16], 10
        jne     .L6
        add     DWORD PTR [rbp-12], 1
        jmp     .L6
```

```
switch(c){
  case ' ':
    ++ns;
    break;
  case '\t':
    ++nt;
    break;
  case '\n':
    ++nl;
}
```

```
        mov     eax, DWORD PTR [rbp-16]
        cmp     eax, 10
        je      .L4
        cmp     eax, 32
        je      .L5
        cmp     eax, 9
        je      .L6
        jmp     .L3
.L5:
        add     DWORD PTR [rbp-4], 1
        jmp     .L3
.L6:
        add     DWORD PTR [rbp-8], 1
        jmp     .L3
.L4:
        add     DWORD PTR [rbp-12], 1
.L3:
        jmp     .L7
```

If versus Switch: Muchos casos

```
int main(void){
  int v=42;
  void f1(void), f2(void), f3(void),
  f4(void), f5(void), f6(void);

  if(v == 1) f1();
  else if(v == 2) f2();
  else if(v == 3) f3();
  else if(v == 4) f4();
  else if(v == 5) f5();
  else if(v == 6) f6();
}
```

```
compare value, 1
   jump if not equal label2
   call function1
   jump exit
label2:
   compare value, 2
   jump if not equal label3
   call function2
   jump exit
label3:
   compare value, 3
   jump if not equal label4
   call function3
   jump exit
label4:
   compare value, 4
   jump if not equal label5
   call function4
   jump exit
label5:
   compare value, 5
    jump if not equal label6
   call function5
    jump exit
label6:
   compare value, 6
   jump if not equal exit
   call function6
```

```
   cmp  dword ptr [rbp - 8], 1
   jne  LBB0_2
   call _f1
   jmp  LBB0_17
LBB0_2:
   cmp  dword ptr [rbp - 8], 2
   jne  LBB0_4
   call _f2
   jmp  LBB0_16
LBB0_4:
   cmp  dword ptr [rbp - 8], 3
   jne  LBB0_6
   call _f3
   jmp  LBB0_15
LBB0_6:
   cmp  dword ptr [rbp - 8], 4
   jne  LBB0_8
   call _f4
   jmp  LBB0_14
LBB0_8:
   cmp  dword ptr [rbp - 8], 5
   jne  LBB0_10
   call _f5
   jmp  LBB0_13
LBB0_10:
   cmp  dword ptr [rbp - 8], 6
   jne  LBB0_12
   call _f6
```

Branch Table ó Jump Table

```
int main(void){
  int v=42;
  void f1(void), f2(void), f3(void),
  f4(void), f5(void), f6(void);

  switch(v) {
    case 1: f1(); break;
    case 2: f2(); break;
    case 3: f4(); break;
    case 5: f5(); break;
    case 6: f6(); break;
  }
}
```

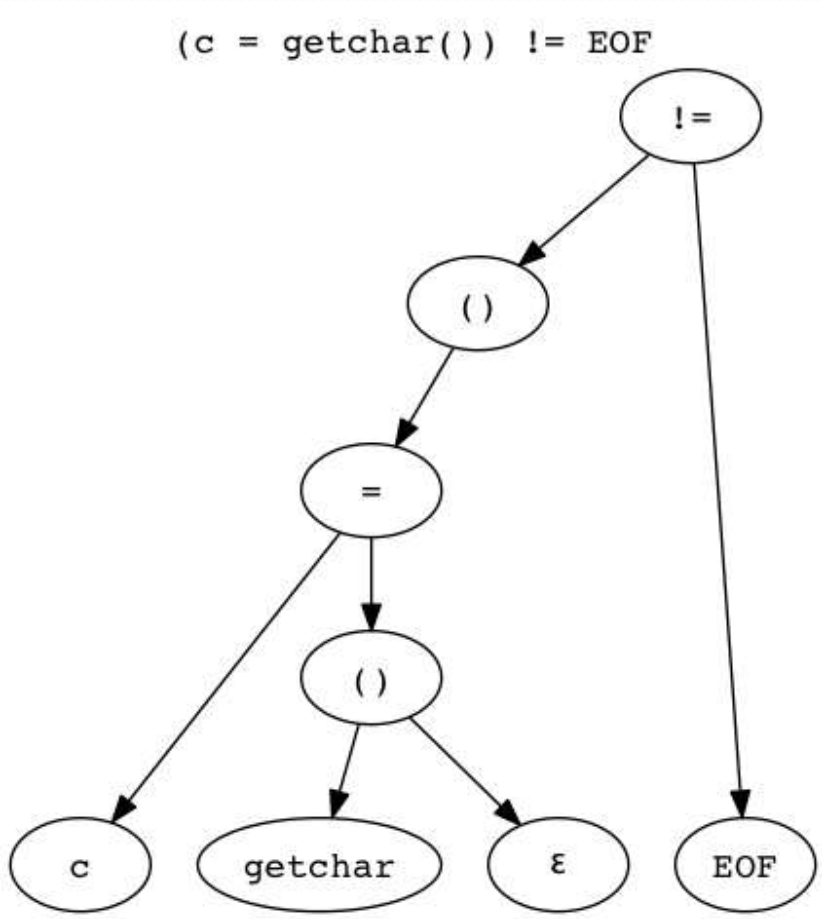
```
    jump table[value]
Label1:
    call function1
    jump exit
Label2:
    call function2
    jump exit
Label3:
    call function3
    jump exit
// Table
                1: Label1
                2: Label2
                3: Label3
```

```
add program_counter, value
call function1
call function2
call function3
```

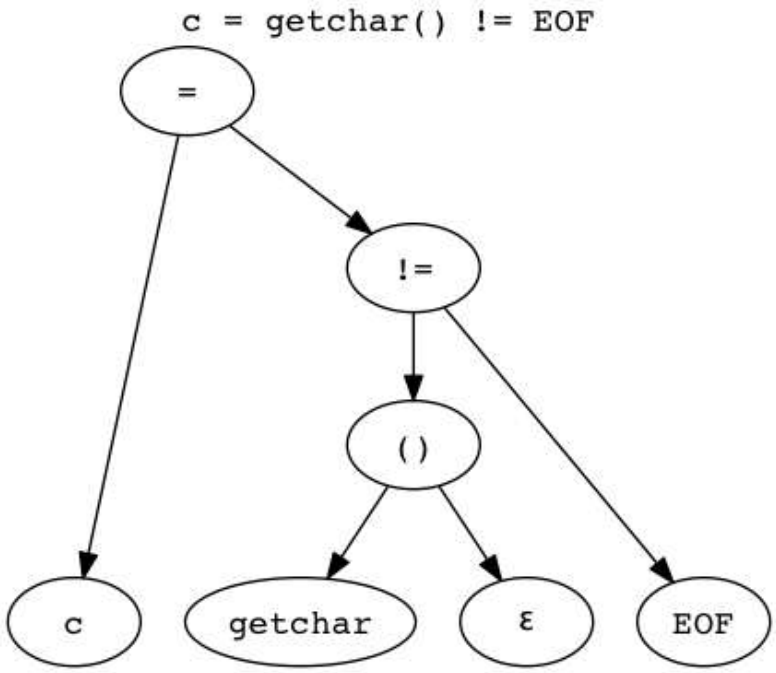
```
    lea   rax, [rip + LJTI0_0]
    mov   rcx, qword ptr [rbp - 16]
    movsxd  rdx, dword ptr [rax + 4*rcx]
    add   rdx, rax
    jmp   rdx
LBB0_1:
    call  _f1
    jmp   LBB0_6
LBB0_2:
    call  _f2
    jmp   LBB0_6
LBB0_3:
    call  _f4
    jmp   LBB0_6
LBB0_4:
    call  _f5
    jmp   LBB0_6
LBB0_5:
    call  _f6
LBB0_6:
    ...
L0_0_set_1 = LBB0_1-LJTI0_0
L0_0_set_2 = LBB0_2-LJTI0_0
L0_0_set_3 = LBB0_3-LJTI0_0
L0_0_set_6 = LBB0_6-LJTI0_0
L0_0_set_4 = LBB0_4-LJTI0_0
L0_0_set_5 = LBB0_5-LJTI0_0
LJTI0_0:
    .long L0_0_set_1
    .long L0_0_set_2
    .long L0_0_set_3
    .long L0_0_set_6
    .long L0_0_set_4
    .long L0_0_set_5
```

Árboles de Expresión

Precedencia



- Abstracción del control de ejecución
  - En sentencias
    - Secuencia
    - Selección
    - Iteración
  - En expresiones
    - Órden de ejecución de operación: Precedencia de operadores
    - Óden de evalación de operandos (más adelante)
- Aridad de los operadores
- Uso de paréntesis
  - Invocación: f()
  - Agrupación: (a+b)*c
  - Delimitación: while(e)s



Introducción a Máquinas de estado

Contador de Palabras

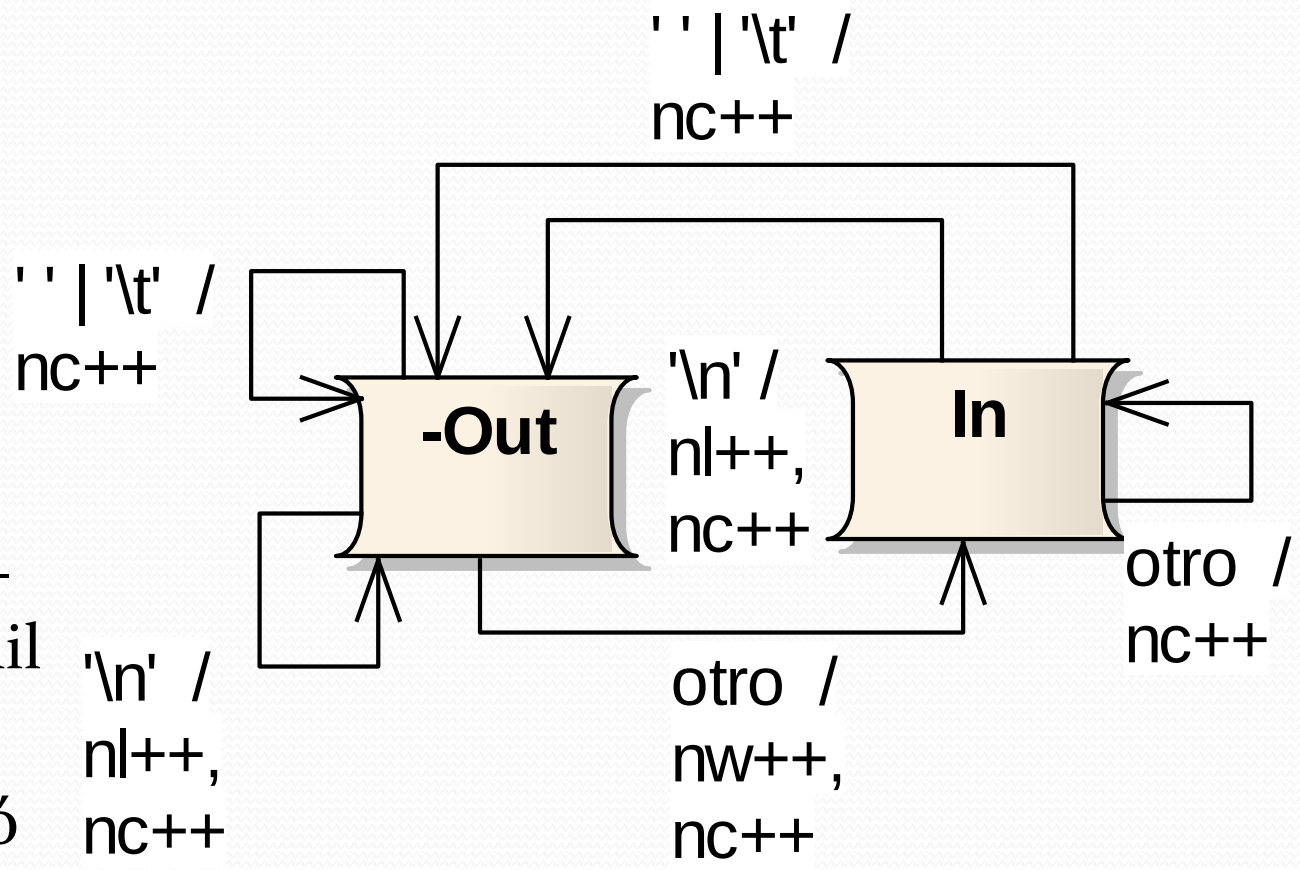
K&R 1.5.4 Conteo de Palabras

Problema

- Contar palabras, líneas y caracteres
- ¿Qué es una palabra?
- Diseño lógico
- Tecnología de implementación
- Abstracciones disponibles
  - Paradigma tipo imperativo
    - Paradigma procedural
      - Estilo estructurado
      - Estilo lineal, sin estructura

Solución

- Máquina de estados con acciones
  - Diagrama de transiciones – Notación Símil UML
- Psuedocódigo ó
- Algoritmo.



Implementaciones de Máquinas de Estado

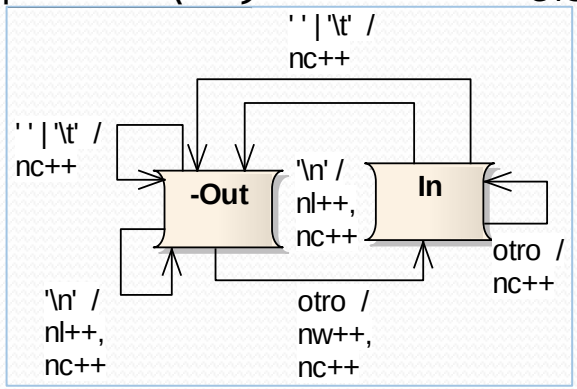
Implementación y Generalización

```c
#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;
}
```




```c
#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)
      if (state == OUT )
         if (c == '\n'){
            ++nl, ++nc;
            state = OUT;
         }
         else if (c==' ' || c=='\t'){
            ++nc;
            state = OUT;
         }
         else {
            ++nw, ++nc;
            state = IN;
         }
      else /* IN */
         if (c == '\n'){
            ++nl, ++nc;
            state = OUT;
         }
         else if (c==' ' || c=='\t'){
            ++nc;
            state = OUT;
         }
         else {
            ++nc;
            state = IN;
         }

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

Términos de la clase #07

Definir cada término con la bibliografía

- If versus Switch
  - Sentencias de Salto
  - Sentencias etiquetadas
  - Branch table ó Jump table
  - Bytes como datos o como instrucciones
  - Eficiencia en tiempo y en espacio
  - Cuando considerar la eficiencia
- Árboles de Expresión
  - Abstracción del control de ejecución en expresiones
  - Precedencia de Operadores
  - Hojas
  - Raíz
  - Nodos no hoja
  - Uso de los paréntesis
  - Invocación
  - Agrupación
  - Delimitación
- Máquinas de Estado
  - Acciones
  - Notación UML para máquina de estados
  - Estado inicial
  - Transición
  - Implementación #1
  - Estados como variable entera
  - Transiciones como selección estructurada y actualización de variable
  - Implementación #1 variante if
  - Implementación #1 variante switch
  - Estado como función
  - Estado como etiqueta.

Tareas para la próxima clase

1. Trabajo #2 – Contador
   - Trabajo opcional, no obligatorio
   - Árboles de Expresión
   - Implementaciones de máquinas de estado.

¿Consultas?

Fin de la clase