

Clase #08 de 27

Máquinas de Estado II, Punteros, y Trabajo #3 Remover Comentarios

Mayo 8, Lunes

Repaso Clase Anterior

- [Glosario](#)
- [Tareas](#)

Agenda para esta clase

- Repaso y árboles de expresión del trabajo #2
- Máquinas de Estado II – Implementación
- Implementación #2
- Strings, Arrays, Punteros, Pre y Pos Incremento
- Trabajo #3 – Remover Comentarios

Repaso

- Token
- Lexema
- Tipos de Token
 - Identificadores
 - Literales Carácter
 - Literales Cadena
- If versus Switch
- Máquinas de Estado
 - Implementación #1
 - Estados como variable entera
 - Transiciones como selección estructurada y actualización de variable
 - Implementación #1 variante if
 - Implementación #1 variante switch
- Construir árboles de expresiones de
 - $n_l = n_w = n_c = 0$
 - $c == ' ' || c == '\n' || c == '\t'$



Intervalo

20 minutos

Otras Implementaciones de Máquinas de Estado

Implementación #1 – Estados como variable entera y Transiciones como selección estructurada y actualización de variable

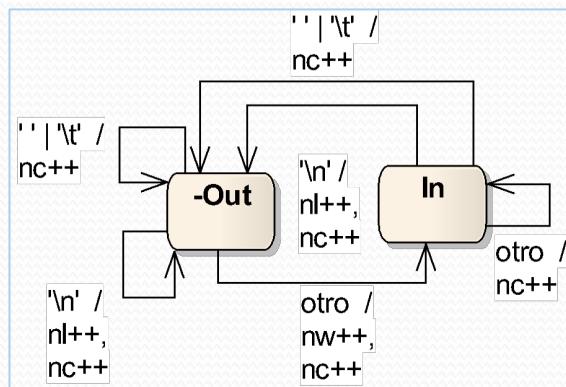
- Algoritmo de la implementación #1 para “máquina de Mealy”
 - Mientras haya caracteres
 - Seleccionar Estado
 - Seleccionar Carácter
 - Accionar
 - Actualizar Estado
- ¿Cuál es la abstracción aplicada?
 - Abstracción de datos
 - Abstracción del control de flujo
- Implementación #1
 - Estado como **variable entera**
 - Transiciones como **selección estructurada y actualización de variable**
- ¿Con qué estructura de control de flujo la implementaría? ¿Con if o con switch? ¿Por qué?
- Eficiencia de If y de Switch
 - Secuencia de comparaciones
 - Tabla de salto
- Implementación #1 con if
- Implementación #1 con switch

Implementación #1 – If versus Switch

```

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;

```



```

while ((c = getchar()) != EOF)
switch(state){
    case OUT:
        switch(c){
            case '\n':
                ++nc, ++nl;
                state = OUT;
                break;
            case ' ':
            case '\t':
                ++nc;
                state = OUT;
                break;
            default:
                ++nc;
                state = IN;
        }
        break;
    default:///* IN */
        switch(c){
            case '\n':
                ++nc, ++nl;
                state = OUT;
                break;
            case ' ':
            case '\t':
                ++nc;
                state = OUT;
                break;
            default:
                ++nc;
                state = IN;
        }
}
printf("%d %d %d\n", nl, nw, nc);
return 0;

```

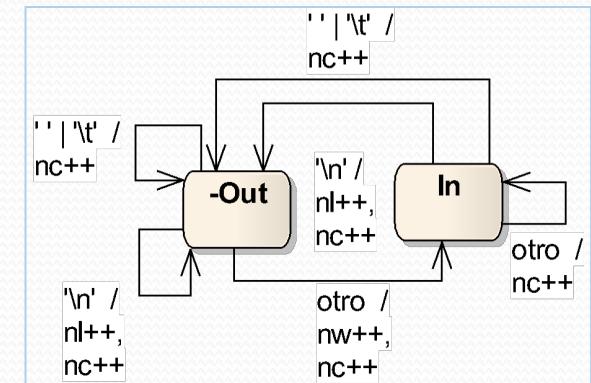
Más Implementaciones

- Implementación #1
 - Estados como **variable enum**
 - Transiciones como **selección estructurada y actualización de variable**
- Implementación #2
 - Estados como **etiquetas**
 - Transiciones como **selección estructurada y saltos incondicionales**
- Implementación #3
 - Estados como **funciones recursivas**
 - Transiciones como **selección estructurada e invocaciones**
- Implementación #4
 - Estado como **variable entera**
 - Transiciones como **tabla**
 - Matriz de pares
 - Arreglo de arreglo de estructuras de dos campos
- Implementación #5
 - Estados como **funciones y variable puntero a función actual**
 - Transiciones como **selección estructurada, actualización de variable e invocaciones**
- Implementación #6
 - Estados como **variable enum**
 - Transiciones como **función GetNextState(s,c) y DoAction(s) ó DoAction(s,c)**
- Otras...

Implementación #2 – Estados como etiquetas y transiciones como saltos

```
int main(void){  
    int nl, nw, nc;  
  
    nl = nw = nc = 0;  
    goto Out;  
  
Out:  
    switch( getchar() ){  
        case '\n':  
            ++nl, ++nc;  
            goto Out;  
        case ' ':  
        case '\t':  
            ++nc;  
            goto Out;  
        case EOF:  
            goto End;  
        default:  
            ++nw, ++nc;  
            goto In;  
    }  
}
```

```
In:  
    switch( getchar() ){  
        case '\n':  
            ++nl, ++nc;  
            goto Out;  
        case ' ':  
        case '\t':  
            ++nc;  
            goto Out;  
        case EOF:  
            goto End;  
        default:  
            ++nc;  
            goto In;  
    }  
  
End:  
    printf("%d %d %d\n", nl, nw, nc);  
    return 0;
```



Ejercicios de K&R que se resuelven con Máquinas de Estado

- 1-12. Escriba un programa que imprima su entrada de a una palabra por línea
- 1-9. Escriba un programa que copie su entrada en su salida, remplazando cada secuencia de uno o más blancos por un solo blanco
 - Diseñar con máquina de Mealy
 - Diseñar con máquina de Moore
- 1-23. Escriba un programa que remueva todos los comentarios de un programa C. No se olvide de tratar correctamente las cadenas y los caracteres literales. Los comentarios en C no se anidan.

Strings, Arrays, Punteros, Pre y Pos Incremento

<https://josemariasola.github.io/ssl/papers/ArraysPointersPrePosIncrement.pdf>

Trabajo #3

Remover comentarios

Trabajo #3 – Remover comentarios

[K&R1988] 1-23

- Basado en ejercicio 1-23 [K&R1988]:
"Escriba un programa que remueva todos los comentarios de un programa C. Los comentarios en C no se anidan. No se olvide de tratar correctamente las cadenas y los caracteres literales"
- Alcance
 - Máquina de estado híbrida o pura: Mealy y Moore
 - Diagrama de estados
 - Formalización
 - Indicar secciones Mealy y secciones Moore
 - Dos implementaciones, descriptas. Ninguna puede ser la Implementación #1: estado como variable y transiciones con selección estructurada.
 - Si es aplicable, utilizar enum
 - Switch
- Productos
 - Diagrama de máquina de estados
 - Diagrama de transiciones para un autómata finito de cada lenguaje
 - Expresión regular para cada lenguaje
 - Implementación 1
 - Implementación 2
 - Bechmark
 - Makefile

Términos de la clase #08

Definir cada término con la bibliografía

- Repaso
 - Token
 - Lexema
 - Identificadores
 - Literales Carácter
 - Literales Cadena
 - If versus Switch
 - Implementación #1 de máquina de estado
 - Árbol de Expresión
- Implementaciones de Máquinas de Estado
 - Implementación #2
 - Estados como etiquetas
 - Transiciones como saltos incondicionales
 - Sentencia etiquetada
 - Sentencia de salto
 - Sentencia goto
 - Implementación #1
 - Estados como variable entera
 - Transiciones como selección estructurada y actualización de variable
 - Implementación #2
 - Estados como etiquetas
 - Transiciones como selección estructurada y saltos incondicionales
 - Introducción a Implementación #3
 - Estados como funciones recursivas
 - Transiciones como selección estructurada e invocaciones
 - Introducción a Implementación #4
- Estado como variable entera
- Transiciones como tabla
 - Matriz de pares
 - Arreglo de arreglo de estructuras de dos campos
- Introducción a Implementación #5
 - Estados como funciones y variable puntero a función actual
 - Transiciones como selección estructurada, actualización de variable e invocaciones
- Introducción a Implementación #6
 - Estados como **variable enum**
 - Transiciones como **función GetNextState(s,c)** y **DoAction(s)** ó **DoAction(s,c)**
- Strings, Arrays, Punteros, Pre y Post Incremento
 - Representación de strings
 - Repaso de efecto de lado de una expresión
 - Repaso de valor de una expresión
 - Valor del preincremento
 - Valor del postincremento
 - Valor de una arreglo: dirección de memoria del primer elemento
 - Aritmética de punteros
 - Incremento de puntero
- Trabajo #3 – Removedor de Comentarios
 - Repaso de secuencia de escape en caracteres y cadenas
 - Comentarios de múltiples líneas
 - Comentarios de una línea

Tareas para la próxima clase

1. Comenzar a diseñar la máquina de estados que remueve comentarios.



¿Consultas?



Fin de la clase

Clase #9 de 27

Especificación Máquinas de Estado

Mayo 15, Lunes

Repaso Clase Anterior

- [Glosario](#)
- [Tareas](#)

Agenda para esta clase

- Arreglos y Expresiones
- Introducción a Cadenas
- Variables Externas
- Recursividad
- Implementación Recursiva de Máquina de Estado
- Tokens

Arreglos y Expresiones

K&R 1.6 Arreglos

Contar dígitos, blancos y otros

(1 de 5)

- Programa que cuente la cantidad de ocurrencias de cada dígito, blanco (espacio, tab, nuevalínea), y todo otro carácter
- Salida
 - dígitos = 9 3 0 0 0 0 0 0 0 1, white space = 123, other = 345
- 12 categorías en entrada; aplicar abstracción de datos:
 - otros
 - nblancos
 - ndigitos(i)
 - No diez variables
- Implementación: Arreglo de enteros
- Estructura del programa
 - Inicialización
 - Clasificación de la entrada
 - Salida.

Contar dígitos, blancos y otros (2 de 5)

```
#include <stdio.h>

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

    /* Inicialización */

    nwhite = nother = 0;

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

- "*La declaración sigue el uso*"
- Base y Offset
- Operación subindicación
 - Binaria
 - Tipos
- Asociatividad
 - ID
 - DI
- Idiom para recorrer arreglo.

Contar dígitos, blancos y otros

(3 de 3)

```
/* Clasificación de la entrada */

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

- '1' versus 1
- Adyacentes
- Subíndice: entero
- Carácter literal es un entero tipo int
- Precedencia de Operadores
- Corto circuito (short-circuit) y orden de evaluación
 - $x \&& y$
 - if x then y else false
 - $x || y$
 - if x then true else y
- Estilo de Multiway-if.

Árbol de Expresiones

- Expresión
- Operador
- Operando
- Valor
- Tipo
- Efecto de lado
- Orden de evaluación de los operandos
- Precedencia de ejecución de las operaciones
- Asociatividad (agrupacion)
- Expresión Primaria

```
c >= '0' && c <= '9'  
c==' ' || c=='\n' || c=='\t'
```

Contar dígitos, blancos y otros (5 de 5)

```
/* Salida */

printf("digits =");

for (i = 0; i < 10; ++i)
    printf(" %d", ndigit[i]);

printf(", white space = %d, other = %d\n",
       nwhite, nother);

return 0;
}
```

Ejercicios

- 1-13. Escribir un programa para imprimir el histograma de la longitud de las líneas (palabras) en su entrada. Más complejo, orientación vertical
- 1-14. Escribir un programa para imprimir el histograma de las frecuencias de los diferentes caracteres en su entrada.



Cadenas

K&R 1.9 Cadenas

Cadenas en ANSI C

- ¿En Pascal?
- ¿En Turbo Pascal?
- ¿En C++?
- ¿En Smalltalk?
- ¿En ANSI C?
 - ¿En el Lenguaje?
 - ¿En la Biblioteca?
 - Representación

¿Por qué C no tiene el tipo String?

- ¿No tiene strings?
- Los arreglos no son "*first class citizens*"
- El tamaño del arreglo no es parte del tipo
 - Contrastar
 - Pascal
 - Java/C#
 - C++
 - Smalltalk
- No hay una representación directa en los registros, no hay operaciones en Lenguaje Máquina para manejar strings

Arreglos de Caracteres

- Ejemplo: Programa que imprime la línea más larga
- Pseudocódigo:
 - Mientras haya más líneas:
 - Si es más larga que la anterior:
 - Guardar esa línea y su longitud
 - Imprimir la línea más larga
 - Diseño Top-Down
 - Obtener una línea de la entrada
 - Guardar una línea
 - Funciones
 - GetLine
 - Copy.

Obtener una línea de la entrada

- Datos
 - Flujo
 - Resultados
 - ¿Línea o Cadena?
 - ¿Flujo?
 - ¿Longitud?
 - ¿Éxito?
 - Función matemática
 - GetLine :
 - Flujo →
 - Flujo × Cadena × B × N
 - ¿‘\n’?
 - Parámetros inout
 - ¿Máximo?
- Un mismo resultado para señalizar dos situaciones diferentes. Similar a getchar
 - Prototipo ANSI C

```
int  
getline(  
char line[],  
int maxline); .
```

Copiar una cadena

- Datos
 - Cadena
- Resultados
 - Cadena
- Función matemática
 - Copiar : Cadena → Cadena / $\text{Copiar}(x)=x$
- Parámetros inout
- Precondiciones
- Prototipo ANSI C
 - void copy(char to[], char from[]);

Imprime la línea más larga

```
#include <stdio.h>

/* maximum input line length */
const int MAXLINE=1000;

/* Verbos */
int getline(char line[], int maxline);
void copy(char to[], char from[]);

int main(void){
    int len; /*current line length */
    int max; /*maximum length seen so far*/

    /* Sustantivos */
    char line[MAXLINE]; /*current line*/
    char longest[MAXLINE];/*saves longest*/
```

- Verbos para funciones
- Sustantivos para variables
- Comunicación
GetLine – Main
- Almacenamiento
(storage) para arreglos

Imprime la línea más larga (cont.)

```
max = 0;  
  
while((len=getline(line,MAXLINE)) > 0)  
  if (len > max) {  
    max = len;  
    copy(longest, line);  
  }  
  
if (max > 0) /*there was a line*/  
  printf("%s", longest);  
  
}
```

- Copy es void,
procedimientos de
Pascal
- %s espera String
- Literal "abc\n"
- | a | b | c | \n | \o |
- ¿Línea demasiado
larga?
 - GetLine
 - Copy
 - Reescribir Main.

Imprime la línea más larga (cont.)

```
/* getline: read a line into s, return length */
int getline(char s[], int lim){
    int c, i;

    for(i=0;
        i < lim-1 && (c=getchar())!=EOF && c!='\n';
        ++i
    )
        s[i] = c;

    if (c == '\n') {
        s[i] = c;
        ++i;
    }

    s[i] = '\0';

    return i;
}
```

- ¿Precondiciones?
- ¿Poscondiciones?
- Valor 반환ado para señalizar evento (EOF) y valores posibles de i
- ¿Próxima lectura?
- ¿Qué ocurre con líneas mayores a lim?
- Cerrar cadena con EOS ('\0').

Imprime la línea más larga (cont.)

```
/* copy: copy 'from' into 'to';
   assume 'to' is big enough */
void copy(char to[], char from[]){
    int i;

    i = 0;
    while( (to[i] = from[i]) != '\0' )
        ++i;
}
```

- ¿Precondiciones?
- ¿Poscondiciones?
- Se usa por su efecto, no por su valor.
Entonces void
- Precedencia
- Valor de la asignación
- Caso límite
- Cerrar cadena con EOS ('\0').

Ejercicios

- 1-16. Revise la función main del programa “línea más larga” para que imprima correctamente la longitud de líneas arbitrariamente largas, y para que imprima la mayor cantidad de texto posible para esa línea
- 1-17. Escriba un programa que imprima las líneas mayores a 80
- 1-18. Escriba un programa que saque los blancos del final de cada línea y elimine las líneas en blanco
- 1-19. Escriba una función reverse(s) que invierte la cadena s. Úselo en un programa que invierta su entrada de a líneas.



Intervalo

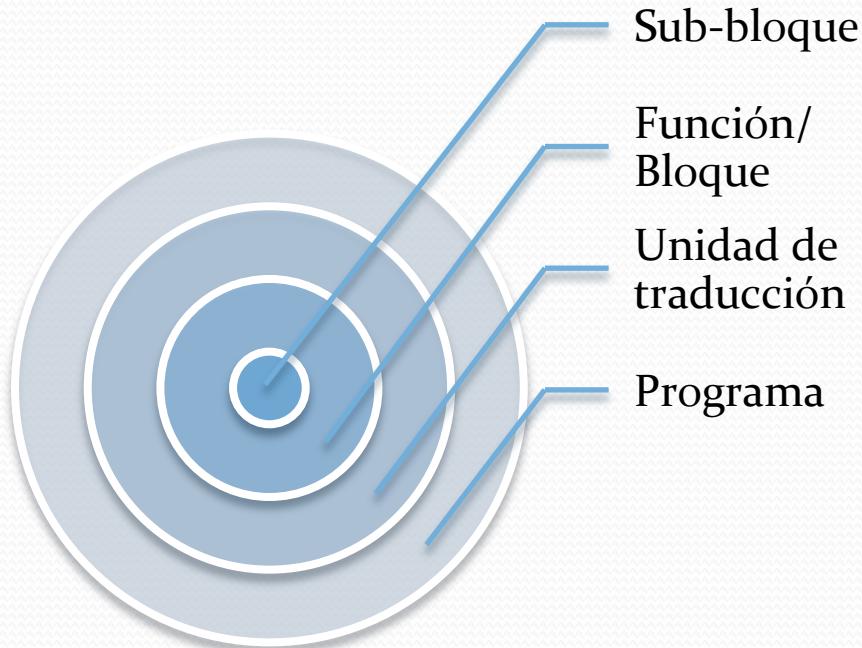
20 minutos

Variables Externas y Alcance

K&R 1.10 Variables Externas y Alcance

Última parte de Introducción a ANSI C

Alcance y Vinculación



- Programa
 - ¿Qué es un programa?
 - Conjunto de Unidades de Traducción (UT, archivos fuente .c)
 - Vinculación entre todas las UT
 - o entre solo algunas UT
 - o entre algunas funciones de algunas UT
- UT
 - Alcance"entre todas las funciones
 - o vinculación entre solo algunas funciones
- Función
 - Alcance en el bloque del cuerpo que la define
- Sub Bloque
 - Alcance en el subbloque.

Variables Automáticas versus Variables Externas

VARIABLES AUTOMÁTICAS

- Variable Automática:
privada o local
 - Solo la función tiene acceso directo
 - ¿Parámetros?
 - Son variables automáticas
- ¿Por qué automáticas?
 - Reserva y liberación de forma automática
 - Push de argumentos
 - Pop en parámetros
 - Comienzan al invocar la función
 - Desaparecen al terminar
 - No retienen valor entre invocaciones
 - Sin inicialización ⇒ Basura
- Mismo Nombre \neq Mismo Objeto
 - Variable i de getline y de copy

VARIABLES EXTERNAS

- Definidas externas a toda función:
Públicas o Globales
- Acceso directo, por nombre, desde "todas" las funciones de la unidad de traducción
 - Antes de usar un nombre de variable, debe estar declarado
 - Comparar con el bloque más externo de Pascal
- Alternativa a lista de argumentos para comunicación entre funciones
- Definición y Declaración
 - Definida una sola vez, de forma externa ⇒ Reserva de memoria
 - Declaradas (tipo de dato) en cada función que la usa
 - De forma Explícita con keyword extern
 - De forma Implícita por contexto y orden
- Vinculadas con otras
 - El Linker conecta las variables
 - ¿Entre otras unidades de traducción o en una misma unidad de traducción?
- Son variables estáticas
 - Permanentes ⇒ Retienen el valor
 - Inicialización en cero.

Duración, Alcance y Vinculación

Lifetime, Scope y Link

- Duración (Duration)
 - Estática
 - Eterna
 - Keyword static
 - Automática
 - Parámetros
 - Bloque
 - Keyword auto
 - "Alocada"
 - Funciones estándar
 - malloc
 - calloc
 - realloc
 - free
- Alcance (Scope)
 - Desde la declaración
 - En UT
 - En función
 - En bloque
- Retención de valor entre invocaciones
 - Variable declarada dentro de la función, con la keyword static
 - Alcance: bloque
 - Duración: estática
- Vinculación (Linkage)
 - Funciones y variables
 - Entre varias UT
 - Vinculación Externa
 - Dentro de UT
 - Vinculación Interna
 - Sin vinculación
 - struct, enum, typedef, label

Ejemplo – Versión especializada de Imprimir la línea más larga (1 de 5)

```
#include <stdio.h>
#define MAXLINE 1000 /* maximum input line size */

/*Analizar semántica*/
int max; /* maximum length seen so far */
char line[MAXLINE]; /* current input line */
char longest[MAXLINE]; /* longest line saved here */

int getline(void);
void copy(void);
```

- Declaraciones externas de variables
- Prototipos, ya se usaron como declaraciones externas, pueden hacerse privados a las UT con palabra reservada static
- Definición de variables y de funciones.

Ejemplo – Versión especializada de Imprimir la línea más larga (2 de 5)

```
int main(void){  
    int len;  
  
    /* Analizar semántica */  
    extern int max;  
    extern char longest[];  
  
    ...
```

- Declaración externa
 - Fuera de cualquier función
- Declaración con extern
 - usa extern delante
 - Es opcional si existe definición previa
 - Declaraciones con extern en header (.h)
 - Para vincular entre UT
 - Convención y razón del nombre
 - Puede ser una declaración externa o interna
 - Puede ser definición.



Ejemplo – Versión especializada de Imprimir la línea más larga (3 de 5)

```
/* Uso de externas y de automáticas */
max = 0;
while ((len = getline()) > 0)
    if (len > max) {
        max = len;
        copy();
    }

    if (max > 0) /* there was a line */
        printf("%s", longest);

return 0;
}
```

Ejemplo – Versión especializada de Imprimir la línea más larga (4 de 5)

```
int getline(void){  
    int c, i;  
    extern char line[]; /*Analizar semántica*/  
  
    for(  
        i=0;  
        i < MAXLINE - 1 && (c=getchar())!=EOF && c!='\n';  
        ++i  
    )  
        line[i] = c;  
  
    if (c == '\n') {  
        line[i] = c;  
        ++i;  
    }  
  
    line[i] = '\0';  
  
    return i;  
}
```

Ejemplo – Versión especializada de Imprimir la línea más larga (5 de 5)

```
void copy(void){  
    int i;  
  
    /*Analizar semántica*/  
    extern char line[], longest[];  
  
    i = 0;  
    while ((longest[i] = line[i]) != '\0')  
        ++i;  
}
```

Identificar declaraciones, definiciones, variables automáticas y variables externas

```
#include <stdio.h>
#define MAXLINE 1000

int max;
char line[MAXLINE];
char longest[MAXLINE];

int getline(void);
void copy(void);

int main(void){
    int len;

    extern int max;
    extern char longest[];

    max = 0;
    while ((len = getline()) > 0)
        if (len > max) {
            max = len;
            copy();
        }

    if (max > 0)
        printf("%s", longest);

    return 0;
}
```

```
int getline(void){
    int c, i;
    extern char line[];

    for(
        i=0;
        i < MAXLINE - 1 &&
        (c=getchar())!=EOF && c!='\n';
        ++i
    )
        line[i] = c;

    if (c == '\n') {
        line[i] = c;
        ++i;
    }
    line[i] = '\0';

    return i;
}

void copy(void){
    int i;
    extern char line[], longest[];

    i = 0;
    while((longest[i]=line[i])!='\0')
        ++i;
}
```

Utilización de variables externas

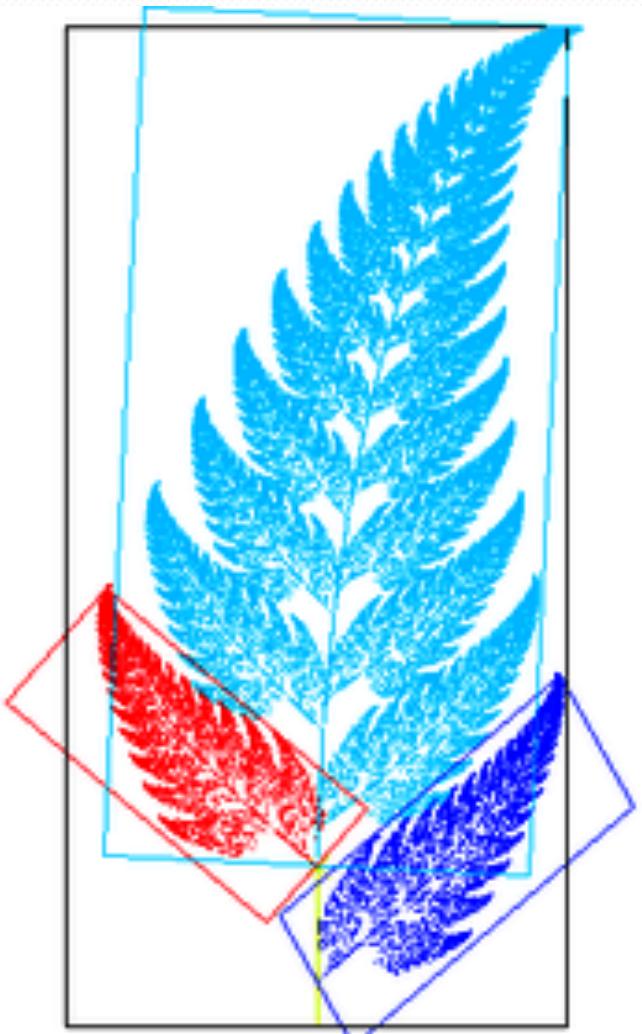
- Pueden ser modificadas accidentalmente
 - Definir al final y declarar solo en las funciones que las deben usar
 - Un programa C puede estar compuesto por varios ó UT: escribir módulos chicos
- Similares a los atributos de instancia o de clase
 - Accesibles por todos los métodos de esa clase
 - En este caso la clase es análoga a la UT
- Las variables externas siempre están disponibles
 - Ventajas
 - Disponibilidad
 - Lista de argumentos más cortas
 - Simplifica comunicación
 - Objetos únicos: stdin, CPU
 - Desventajas
 - Acoplamiento no obvio
 - Bugs
 - Mantenimiento
- En este caso perdemos la oportunidad de crear dos funciones genéricas: getline y copy
 - ANSI provee: fgets y strcpy.

Recursividad

Recursividad

¿Qué es la Recursividad?

- Es la definición de un proceso basado en su propia definición
- Es la repetición de ítems de una manera *autosimilar*
- Alternativa a la iteración
- Definiciones de funciones de manera recursiva
- "*Para entender recursividad, uno debe entender recursividad*"
- GNU



pointer initialization 102, 138
pointer, null 102, 198
pointer subtraction 103, 138, 198
pointer to function 118, 147, 201
pointer to structure 136
pointer, void * 93, 103, 120, 199
pointer vs. array 97, 99–100, 104, 113
pointer-integer conversion 198–199, 205
pointers and subscripts 97, 99, 217
pointers, array of 107
pointers, operations permitted on 103
Polish notation 74
pop function 77
portability 3, 37, 43, 49, 147, 151, 153, 185
position of braces 10
postfix ++ and -- 46, 105
pow library function 24, 251
power function 25, 27
#pragma 233
precedence of operators 17, 52, 95, 131–132,
 200
prefix ++ and -- 46, 106
preprocessor, macro 88, 228–233
preprocessor name, __FILE__ 254
preprocessor name, __LINE__ 254
preprocessor names, predefined 233
preprocessor operator, # 90, 230
preprocessor operator, ## 90, 230
preprocessor operator, defined 91, 232

ptrdiff_t type name 103, 147, 206
push function 77
pushback, input 78
putc library function 161, 247
putc macro 176
putchar library function 15, 152, 161, 247
puts library function 164, 247

qsort function 87, 110, 120
qsort library function 253
qualifier, type 208, 211
quicksort 87, 110
quote character, ' 19, 37–38, 193
quote character, " 8, 20, 38, 194

\r carriage return character 38, 193
raise library function 255
rand function 46
rand library function 252
RAND_MAX 252
read system call 170
readdir function 184
readlines function 109
recursion 86, 139, 141, 1 2, 202, 269
redirection see input/output redirection

Ejemplo – Factorial

$$n! = 1 \times 2 \times 3 \times 4 \times \dots \times (n-1) \times n$$

$$n! = \prod_{k=1}^n k$$

$$n! = \begin{cases} 1 & n = 0 \\ (n-1)! \times n & n > 0 \end{cases}$$

```
def factorial(n):
    r = 1
    for i in range(1,n+1):
        r *= i
    return r
```

```
def factorial(n):
    if n = 0:
        return 1
    return factorial(n-1)*n
```

QuickSort

```
def qsort (arr):  
    less = []  
    pivotList = []  
    more = []  
    if len(arr) <= 1:  
        return arr  
    else:  
        pivot = arr[0]  
        for i in arr:  
            if i < pivot:  
                less.append(i)  
            elif i > pivot:  
                more.append(i)  
            else:  
                pivotList.append(i)  
    less = qsort (less)  
    more = qsort (more)  
    return less + pivotList + more
```

```
qsort []      = []  
qsort (x:xs) = qsort [y|y<-xs, y<x] ++  
              [x] ++  
              qsort [y|y<-xs, y>=x]
```

Tipos de recursividad

- Directa
- Múltiple
- *Tail* (Final)
- Indirecta

Tipos de datos recursivos

```
data List a = Nil | Cons a (List a)
```

```
data Tree a = Leaf a | Branch [Tree a]
```

Recursividad – Pros & Cons

```
void printd(int n){      void qsort(int v[], int left, int right){  
    if (n < 0) {          int i, last;  
        putchar('-');      void swap(int v[], int i, int j);  
        n = -n;            if (left >= right)  
    }                      return; /* do nothing if fewer than two elements */  
    if (n / 10)           /* move partition elem to v[0] */  
        printd(n / 10);  swap(v, left, (left + right)/2);  
    putchar(n%10 + '0');  last = left;  
}  
  
void qsort(int v[], int left, int right){  
    int i, last;  
    void swap(int v[], int i, int j);  
    if (left >= right)  
        return; /* do nothing if fewer than two elements */  
    /* move partition elem to v[0] */  
    swap(v, left, (left + right)/2);  
    last = left;  
    /* partition */  
    for (i = left + 1; i <= right; i++)  
        if (v[i] < v[left])  
            swap(v, ++last, i);  
    swap(v, left, last); /*restore partition elem*/  
    qsort(v, left, last-1);  
    qsort(v, last+1, right);  
}  
  
void swap(int v[], int i, int j){  
    int temp;  
    temp = v[i];  
    v[i] = v[j];  
    v[j] = temp;  
}
```

Ejercicio

- Ejercicio 4-12. Versión recursiva de reverse(s).

Recursividad

Categorías Léxicas

Categorías

- Categorías Léxicas ó Tokens
 - Identificadores
 - Palabras reservadas
 - Literales
 - Enteros
 - Reales
 - Caracteres
 - Cadenas
 - Operadores y puntuación
 - Otros
- ¿Cuántos hay?
- Leer K&R1988
 - A.1
 - A.2
- Leer MUCH2012
 - V1 s3.6.2.

Términos de la clase #09

Definir cada término con la bibliografía

- Arreglos y Expresiones
 - Base y offset de arreglos
 - Declaración de arreglos
 - Declaración sigue al uso en la expresión
 - Operación subindicación de arreglo
 - Asociatividad ID y DI
 - Precedencia de operadores
 - Cortocircuito y orden de evaluación
 - MultiwayMultiway-if
 - Idiom para recorrer arreglo
- Introducción a Strings (Cadenas)
 - Representación en memoria
 - Carácter nulo
 - String literal
- Variables Externas
 - ¿Qué es un programa?
 - Unidad de Traducción (UT)
 - Vinculación entre varias UT
 - Alcance dentro de una UT
 - Vinculación entre diferentes UT
 - Bloque
 - Sub-bloque
 - Variable automática
 - Variable externa
 - Variable estática
 - Vínculo entre variables
- Declaración de variables
 - Definición de variables
 - Alcance ó Alcance Léxico
 - Duración del tiempo de vida o lifetime
 - Estática
 - Automática
 - "Alocada"
 - Declaración Externa
 - Declaración con extern
 - Declaración versus Definción
 - Stack de invocación
 - Inicialización de estáticas
 - Ocultamiento de información
 - Static para encapsular
 - Posincremento ó Incremento (sufijo)
 - Predecremento ó Decremento (prefijo)
 - Aplicación de variables externas.
 - Variable
- Recursividad
- Implementación Recursiva de Máquina de Estado
- Tokens

Tareas para la próxima clase

1. Comenzar a diseñar la máquina de estados que remueve comentarios.



¿Consultas?



Fin de la clase