Clase #23 de 29
Identificadores & Trabajo Traductor de Declaraciones

Oct 7, Martes
Oct 8, Miércoles

Identificadores

Alcance, Espacios de Nombre, Duración, y Enlace

https://josemariasola.wordpress.com/ssl/papers/#Identifiers

Identificadores

- Nivel Léxico
  - Especificación e Implementación
  - Palabras Clave
  - Palabras reservadas
  - Identificadores Predefinidos
- Nivel Semántico
  - Categorías
  - Atributos
  - Scope
  - Namespace
  - Linkage
  - Binding
  - Lifetime
  - Storage Class Duration

Códigos Ejemplo para Analizar

```
void foo();
void foo(){
   enum foo { foo };
   union foo {int foo;};
   struct foo {double foo};

   enum foo foo;

   goto foo;
foo: // espacio de nombre de etiquetas



   return;
}


void foo(){
    //struct foo{int i;};
    //union foo{double d;}
   enum foo { foo };

   enum foo foo;

    goto foo;
foo:
    return;
}

int main(){ void foo(); foo();}

void foo(){
    union  u{ double foo;};
```

```
   struct s{ double foo;};
    enum foo{ foo      };
   enum foo v = foo;
    struct s vs;
    vs.foo=3.14;
    goto foo;
foo:
    return;
}

struct s{double x, y;};
int var;
int bar(int x, int y);
int pos(int x, int y);

int foo(){
   int x;
   goto fin;
   {
      int y;
      goto fin;
   }
fin:
   return;
}


int x;

typedef int Entero;

Entero y;
```

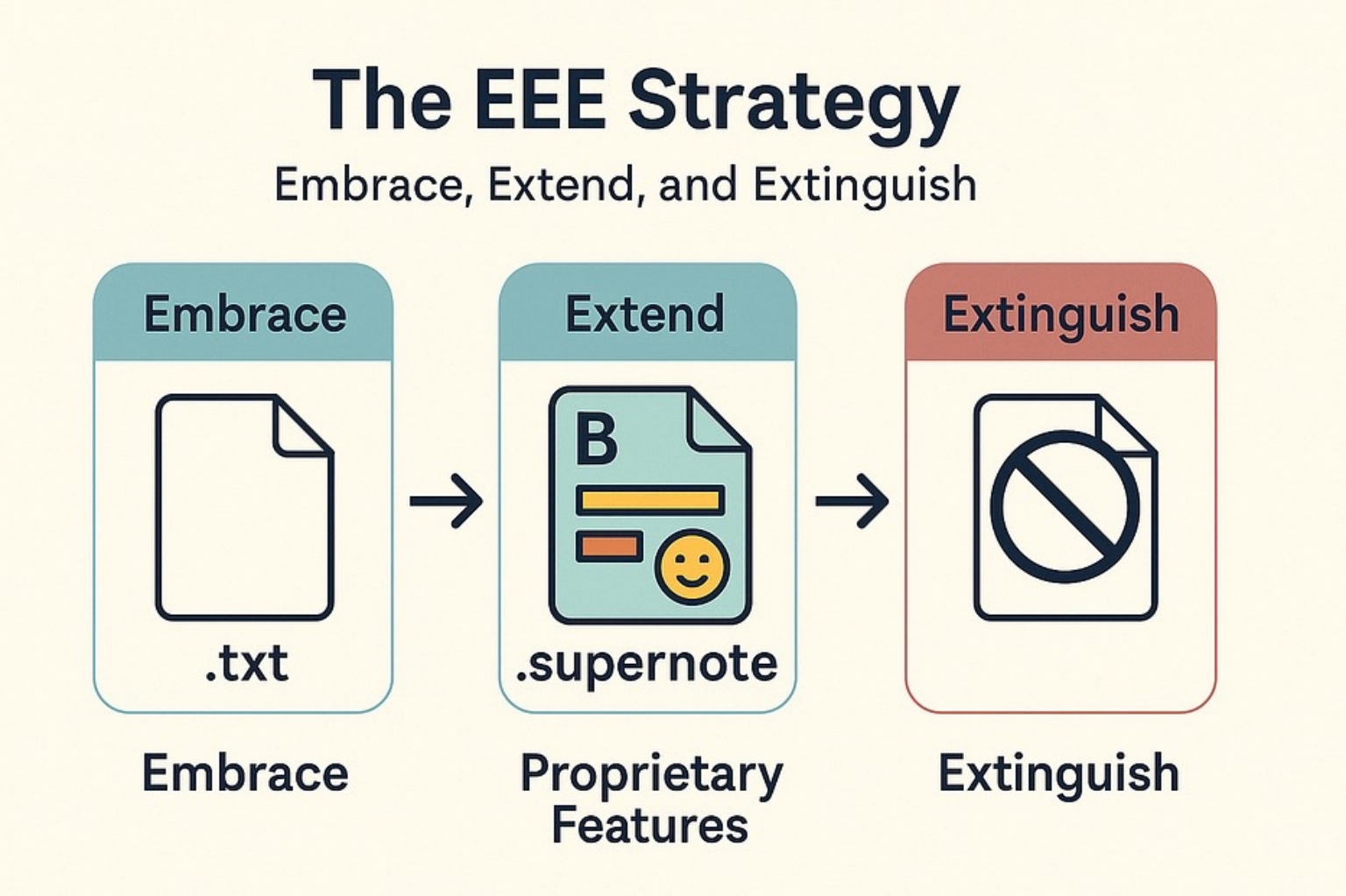
Estrategia: EEE

Internet Explorer

JavaScript

J#

¿Por qué es conveniente tener más de una implementación? Competencia



https://medium.com/@himadri.abm/the-eee-strategy-when-innovation-turns-into-subtle-domination-7cc705347063

Ejemplo de Parser Recursivo Descendente

K&R1988 5.12 Declaraciones Complicadas
MUCH2012 v2s3.2.5 UN PARSER PARA MICRO

Declaraciones – versiones simplificadas

translation-unit :
    declaration
    translation-unit *declaration*

declaration :
    name *dcl* **;**

dcl :
    optional ***** **'** *s direct-dcl*

direct-dcl :
    name
    (*dcl***)**
    direct-dcl **()**
    direct-dcl **[***optional size***]**

Implementación en C

- Funciones para cada variable
- Analizador Sintáctico Descendente Recursivo (ASDR)

Ejemplos de árboles de derivación

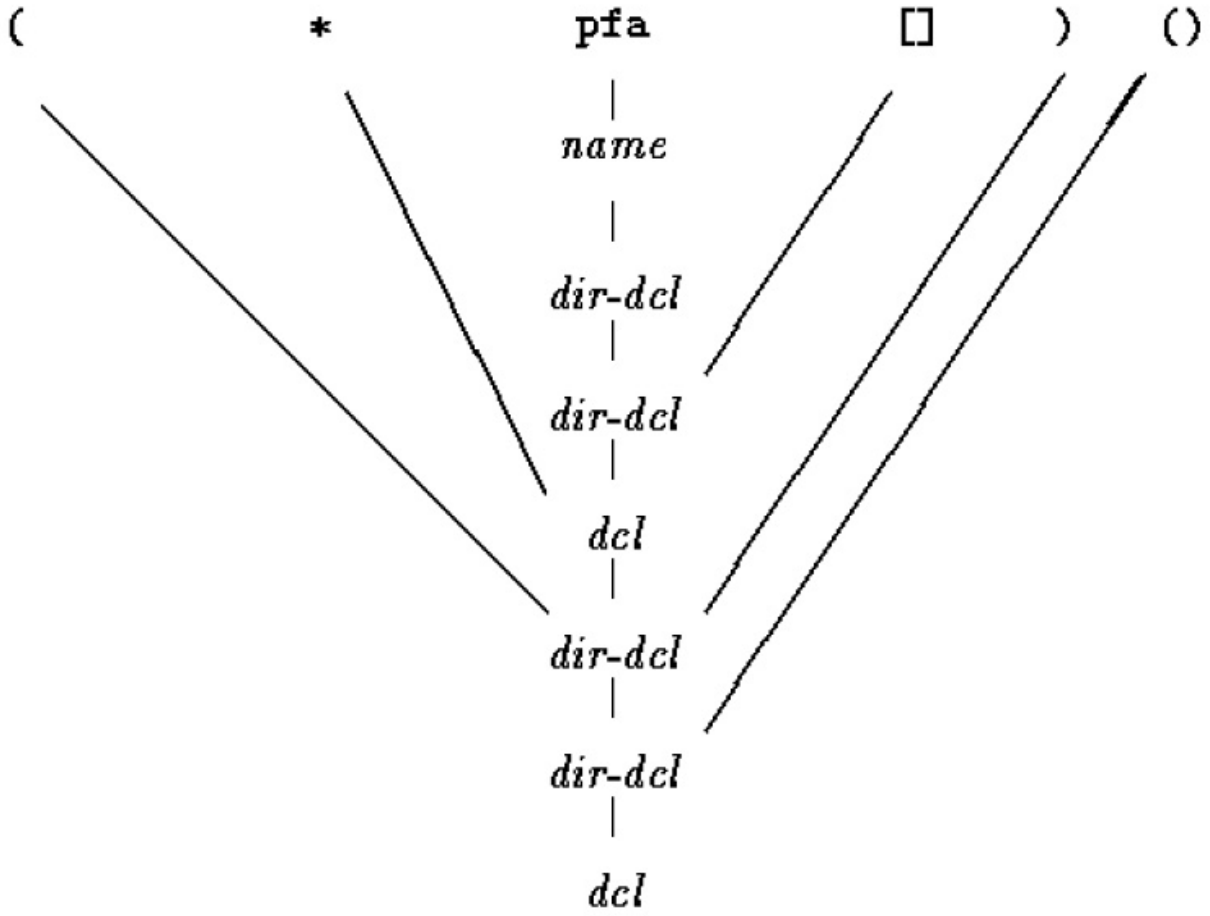
```
int x;

int * y;

int f();

char a[];

float b[7];

long (*p)[9];
```

declaration:
name dcl **;**

dcl:
optional ***** **'s** *direct-dcl*

direct-dcl:
name
(*dcl***)**
direct-dcl **()**
direct-dcl **[***optional size***]**

Ejemplo: (* pfa []) ()



declaration:
name dcl **;**

dcl:
optional ***** **'s** *direct-dcl*

direct-dcl:
name
(*dcl***)**
direct-dcl **()**
direct-dcl **[***optional size***]**

Dcl

```
void dcl(void){

  for (; gettoken() == '*'; )
    ;

  dirdcl();
}
```

declaration:
    name dcl **;**

dcl:
    optional ***'s** *direct-dcl*

direct-dcl:
    name
    (*dcl***)**
    direct-dcl **()**
    direct-dcl **[***optional size***]**

DirDcl

declaration:
name dcl ;

dcl:
optional ***'s** *direct-dcl*

direct-dcl:
name
(*dcl***)**
direct-dcl **()**
direct-dcl **[***optional size***]**

```
void dirdcl(void){
  int type;

  if (tokentype == '(') {
    dcl();
    if (tokentype != ')')
      printf("error: missing )\n");
  } else if (tokentype == NAME)
    ;
  else
    printf("error: expected name or (dcl)\n");

  while ((type=gettoken()) == PARENS || type == BRACKETS)
    if (type == PARENS)
      ;
    else
      ;
}
```

Programa completo - Encabezado

```
#include <stdio.h> /* ungetc */
#include <string.h>
#include <ctype.h>
#define MAXTOKEN 100

enum { NAME, PARENS, BRACKETS };

void dcl(void);
void dirdcl(void);
int gettoken(void);

int tokentype; /* type of last token */
char token[MAXTOKEN]; /* last token string */
char name[MAXTOKEN]; /* identifier name */
char datatype[MAXTOKEN]; /* data type */
char out[1000];
```

Programa completo - Main

```
int main(void){ /* Declaration to words */
  while(gettoken()!=EOF){ /*1st token*/
    strcpy(datatype, token); /*datatype*/
    out[0] = '\0';
    dcl(); /* parse rest of line */
    if (tokentype != '\n')
      printf("syntax error\n");
    printf("%s: %s %s\n",
      name, out, datatype);
  }
  return 0;
}
```

declaration:
name dcl **;**

dcl:
optional ***** **'s** *direct-dcl*

direct-dcl:
name
(*dcl***)**
direct-dcl **()**
direct-dcl **[***optional size***]**

Programa completo – GetToken

```
int gettoken(void) {
   int c;
   char *p = token;

   while ( (c = getchar()) == ' ' || c == '\t' )
      ;

   if (c == '(') {
      if ((c = getchar()) == ')') {
         strcpy(token, "()");
         return tokentype = PARENS;
      } else {
         ungetc(c);
         return tokentype = '(';
      }
   } else if (c == '[') {
      for (*p++ = c; (*p++ = getchar()) != ']'; )
         ;
      *p = '\0';
      return tokentype = BRACKETS;
   } else if (isalpha(c)) {
      for (*p++ = c; isalnum(c = getchar()); )
         *p++ = c;
      *p = '\0';
      ungetch(c);
      return tokentype = NAME;
   } else
      return tokentype = c;
}
```

Dcl con acciones

```
void dcl(void){
  int ns;

  for (ns = 0; gettoken()
== '*'; )
    ns++; /* count *'s */
  dirdcl();
  while (ns-- > 0)
    strcat(out, " pointer
to");
}
```

declaration:
name dcl ;

dcl:
optional ***** *'s direct-dcl*

direct-dcl:
name
(*dcl***)**
direct-dcl **()**
direct-dcl **[***optional size***]**

DirDcl con acciones

direct-dcl:
name
(*dcl***)**
direct-dcl **()**
direct-dcl **[***optional size***]**

```
void dirdcl(void){
 int type;

 if (tokentype == '(') { /* ( dcl ) */
   dcl();
   if (tokentype != ')')
     printf("error: missing )\n");
 } else if (tokentype == NAME) /* variable name */
   strcpy(name, token);
 else
   printf("error: expected name or (dcl)\n");
 while ((type=gettoken()) == PARENS || type == BRACKETS)
   if (type == PARENS)
     strcat(out, " function returning");
   else {
     strcat(out, " array");
     strcat(out, token);
     strcat(out, " of");
   }
}
```

Undcl

```
x () * [] * () char

char (*(*x())[])()
```

```
int main(void){
  int type;
  char temp[MAXTOKEN];
  while (gettoken() != EOF) {
    strcpy(out, token);
    while ((type = gettoken()) != '\n')
      if(type==PARENS || type==BRACKETS)
        strcat(out, token);
      else if (type == '*') {
        sprintf(temp, "(*%s)", out);
        strcpy(out, temp);
      } else if (type == NAME) {
        sprintf(temp, "%s %s",token,out);
        strcpy(out, temp);
      } else
        printf("invalid %s\n", token);
    puts(out);
  }
  return 0;
}
```

Ejercicios

- 5-18. Que dcl se recupere de errores
- 5-19. Modifique undcl para que no agregue paréntesis redundantes a las declaraciones
- 5-20. Expanda dcl para manejar declaraciones con tipos de argumentos, calificadores como const, etc.

Trabajo *dcl*

- Traduje declaraciones de a LN (e.g., castellano, inglés)
- Ejemplo de uso
  - $ dcl < declaraciones.txt
- Especificar
  - Léxico con Regex
  - Sintaxis con BNF
  - Semántica, si aplica, con LN
- Implementa un parser que llama un scanner

Extensiones al Trabajo *dcl*

Posiblemente para recuperatorios, pero no para primera entrega

- Léxicas
  - Symbol table, qsort, bsearch, pre-cargados
  - IsIdentifier, IsKeyword
- Sintácticas
  - Declaradores
    - Varios declaradores separados por coma
    - Lista de parámetros simple, o con id, o con calificadores
    - Inicializadores
  - Declaración
    - Nombres de tipos compuestos, por ejemplo
      - int: signed int
      - unsigned: unsigned int
      - long double
      - Buscar especificación
  - Especificadores de Declaración
    - Especificadores de tipos
      - struct
      - enum
      - union
    - Calificadores de tipo (e.g., const)
    - Especificador de clase de almacenamiento (e.g., static)
- Semánticas
  - Verificaciones de restricciones, por ejemplo
    - Tipo de size
    - Nombre de tipo inexistente
    - Redeclaración

¿Consultas?

Fin de la clase