Clase #25 de 27

Side Effects, L-Value & Rol de Parámetros: In, Out, Inout

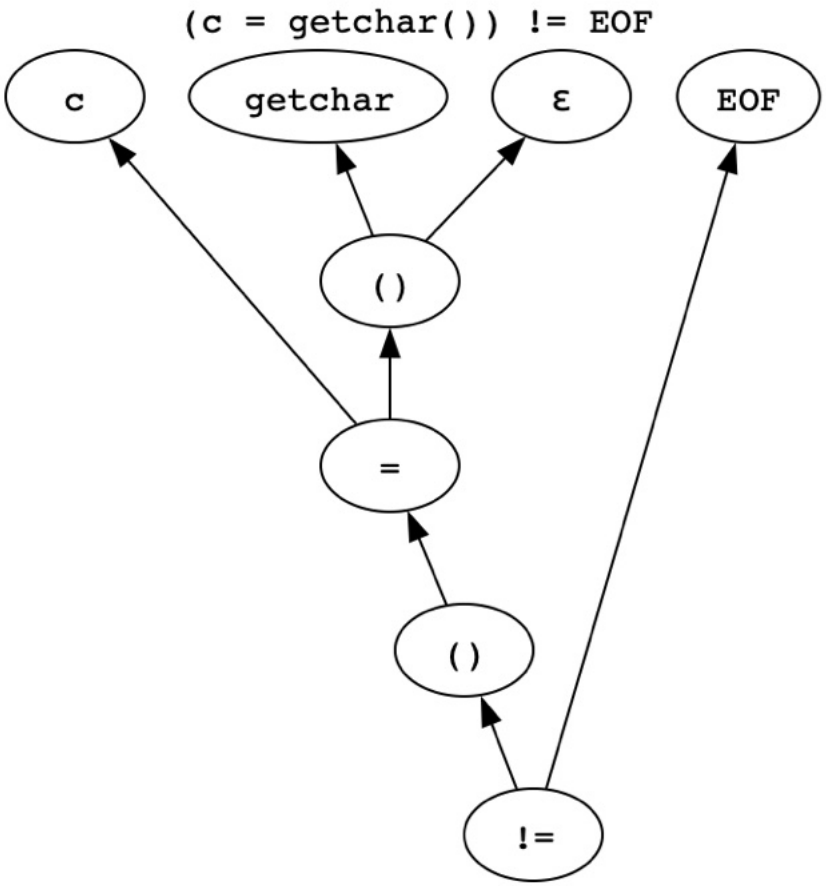
Octubre 21, Martes
Octubre 22, Miércoles

Agenda para esta clase

- Árboles de Expresión
- Side Effects
- Rol de Parámetros: In, Out, Inout

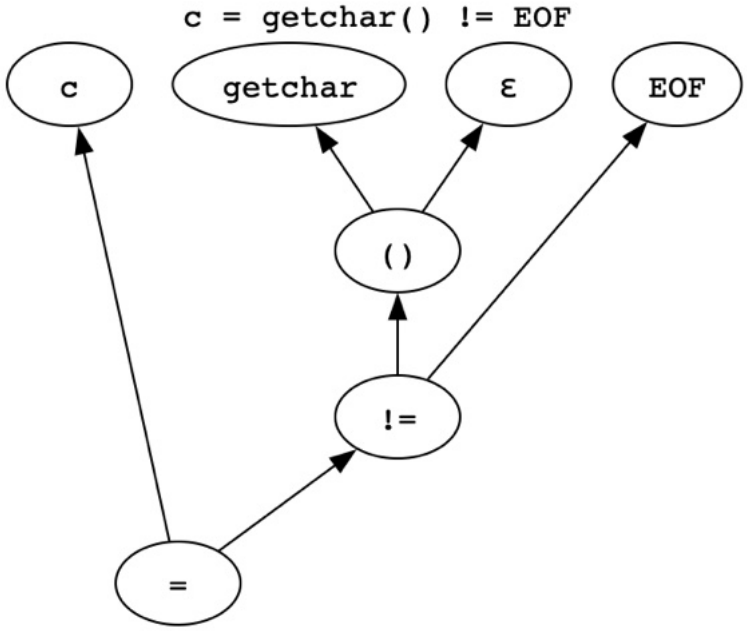
Árboles de Expresión (II)

Precedencia y Efecto de Lado



Abstracción del control de ejecución

- En sentencias
  - Secuencia
  - Selección
  - Iteración
- En expresiones
  - Orden de ejecución de operación: Precedencia de operadores
  - Orden de evaluación de operandos (más adelante)
- Aridad de los operadores
- Uso de paréntesis
  - Invocación: f()
  - Agrupación: (a+b)*c
  - Delimitación: while(e)s
- Reescriba con paréntesis redundantes para denotar precedencia



Valor-L y Restricciones Semánticas

Valor-L y Valor-L Modificable

- Expresión Valor-L
  - Expresión con *tipo objeto* o *tipo incompleto* no **void**
  - Viene de *Left-Value* por su posición en la asignación
  - Pero se lo debe considerar *Valor-de-Localización*
- Expresión Valor-L Modificable
  - Valor-L que es
    - *no tipo arreglo*,
    - no *tipo incompleto*,
      no *tipo calificado const*,
    - y si es tipo unión o estructura, no tiene miembros *tipo calificado const* recursivamente
- SSL 1, p 61, Ejercicio 50
  - Investigue e informe, con ejemplos, qué es un Valor-L modificable

Ejemplo de S&S: Asignación

expresión-de-asignación
expresión-condicional
expresión-unaria *operador-de-asignación* *expresión-de-asignación*

operador-de-asignación uno de
`= *= /= %= += ?= <<= >>= &= ^= |=`

- Restricciones
  - Operando izquierdo es **Valor-L Modificable**
  - Ambos son tipo aritméticos
  - Ambos son tipo estructuras o uniones compatibles
  - Ambos punteros a tipos compatibles
  - Uno puntero void y otro puntero a tipo objeto o a tipo incompleto
  - El izquierdo es un puntero y el derecho es una constante puntero nulo
- Semántica
  - Almacena un valor en el objeto designado por el operador izquierdo
  - El valor de la expresión es el del operando izquierdo, **pero no es un Valor-L**
  - El tipo es del operando izquierda
- Asignación compuesta
  - Equivalencia
    - *E* `=` *E* `op` *E*
    - *E* `op=` *E*
  - ¿Diferencia?
    - Cantidad de Evaluaciones
    - Ejemplo

Restricciones semánticas a las expresiones

- En función del tipo
  - **%**
  - **.**
  - **->**
  - *****
- En función a valor-L
  - **&**
- En función a valor-L modificable
  - **=**
  - **++**
  - etc

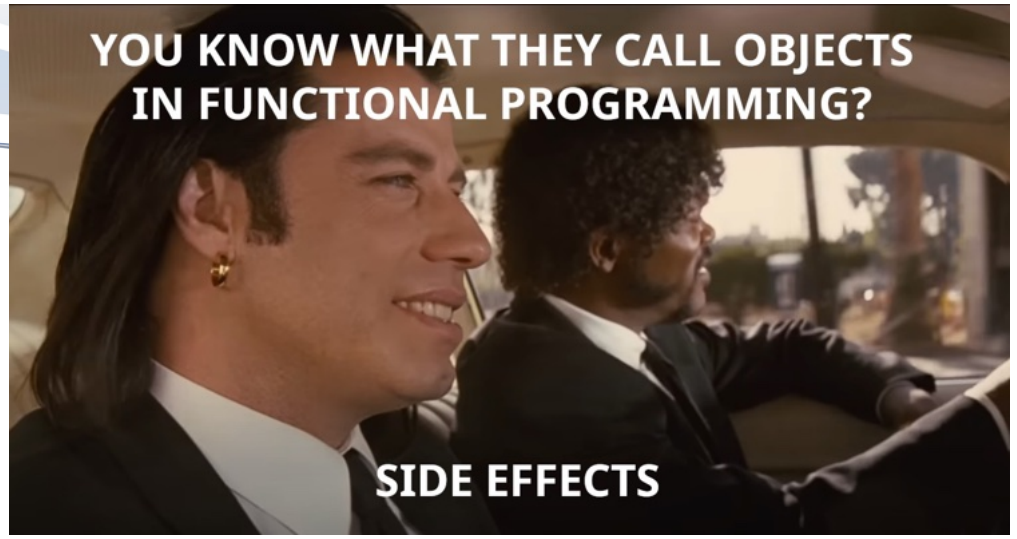
Side Effects

Efecto de lado (traducción literal)

Efecto secundario

Efecto colateral

Efecto de Lado




- ```
  int x;
  x=42;
  x++;
  ```

- ```
  if( (c = *--sp) == '$')
    return state;
  check(c);
  ```

- ```
  getchar();
  ```
- ```
  fgetc(stdin);
  ```

- ```
  send(whatsap_contact, text_message);
  ```
  ```

Operadores con dos Versiones: Con y Sin Efecto de Lado

- Ejercicio 1: Dada la declaración `int i=5;` completar la tabla y determinar
  - tipo,
  - valor,
  - efecto de lado,
  - y categoría (*iso iec 9899 6.3.2.1*)
    - rvalue (o valor de la expresión)
    - lvalue
    - lvalue modificable
  - de cada expresión
- Otros operadores binarios con dos versiones
  - Aritméticos
    - *=
    - /=
    - %=
  - Bitwise
    - Corrimiento
      - <<=
      - >>=
    - Boole
      - &=
      - |=
      - ^=
- Ejercicio 2: Investigar conversión lvalue

Sin efecto			Con efecto			
Expresión	Valor	Tipo y Categoría	Expresión	Valor	Efecto sobre i	Tipo y Categoría
`i+3`			`i+=3`			
`i-3`			`i-=3`			

`i+1`			`i+=1`
			`++i`
			`i++`			

`i-1`			`i-=1`
			`--i`
			`i--`			

Rol de Parámetros

In

Out

Inout

Resumen Rol y Pasaje en C++

Prototipo C++	Invocación C++	Pasaje de Argumento	Rol del parámetro
void f(int x);	***f(1)*** ***int a=2;*** ***f(a)***	Valor (copia)	in
void f(int& x);	***int a=2;*** ***f(a)***	Referencia (variable)	inout out
void f(const int& x);	***f(1)*** ***int a=2;*** ***f(a)***	Referencia (variable) no modificable	in

Esp. Ing. José María Sola, Profesor 422

Transparencia Referencial

- $x = \pi/2$
- $\sin x = 1$

- $\sin x + \sin x = 1 + 1$

- `int getchar();`
- `getchar()`
- `getchar()`
- `getchar()`

- ¿Parámetro implícito?

- `getchar()` equivalente a `fgetc(stdin)`

- `// in out inout`
- `int fgetc(FILE *stream /*inout*/);`

- `fgetc(stdin)`
- `fgetc(stdin)`
- `fgetc(stdin)`

Otros Lenguajes (I)

- C#: `out ref`

```
int h(int energía);
int f(int out energía);
int incrementarEnergía(int ref
energía);

main(){
  int miEnergía;
  f(miEnergía);
  incrementarEnergía(miEnergía)
}
```

- Swift: `inout`
- SQL Stored Procedures:

- Wollok

```
unObjeto.HacéAlgoCon(otroObjeto)

method HacéAlgoCon(otroObjeto){

}

class Persona{
  var prop edad;
  var prop energía;

  method SosMayor() = edad > 21

  method Cená(unPlatoDeComida){
     unPlatoDeComida = unaEnsalada
     energía +=
unPlatoDeComida.Calorías()
     return energía > 1000
  }
}
unaPersona.Cená(milaConFritas)
Cená(unaPersona,milaConFritas)
```

Otros Lenguajes (II)

C

```
struct Persona{
   unsigned edad, energía;
   bool (*SosMayor)();
   unsigned (*Cená)(PlatoDeComida*);
};

struct Persona{
   Pimpl* estado;  // declarado en otro alcance
   bool (*SosMayor)();
   unsigned (*Cená)(PlatoDeComida*);
};

Persona unaPersona;
unaPersona.SosMayor(unaPersona);//bool
unaPersona.SosMayor();          //Define
unaPersona.SosMayor();          //struct offset
unaPersona.Cená(unaMila);
```

C++

```
struct Persona{
      bool SosMayor() const;
      int Cená(PlatoDeComida unPlato);
      int Cená(PlatoDeComida& unPlato);
      int Cená(const PlatoDeComida& unPlato);
   private:
      int energía;
      unsigned edad;
};

class Persona{
      int energía;
      unsigned edad;
   public:
      bool SosMayor() const;
      int Cená(PlatoDeComida unPlato);
      int Cená(PlatoDeComida& unPlato);
      int Cená(const PlatoDeComida& unPlato);
};

unaPersona.SosMayor();
unaPersona.Cená(unaMilaConFritas);
```

Recordar las implementaciones del Stream

FILE Struct en Diferentes Implementaciones

```
// Unix like
typedef struct __sFILE {
    unsigned char *_p; /* current position in (some) buffer */
    int _r; /* read space left for getc() */
    int _w; /* write space left for putc() */
    short _flags; /* flags, below; this FILE is free if 0 */
    short _file; /* fileno, if Unix descriptor, else -1 */
    struct __sbuf _bf; /* the buffer (at least 1 byte, if !NULL) */
    int _lbfsize; /* 0 or -_bf._size, for inline putc */

    /* operations */
    void *_cookie; /* cookie passed to io functions */
    int (* _Nullable _close)(void *);
    int (* _Nullable _read) (void *, char *, int);
    fpos_t (* _Nullable _seek) (void *, fpos_t, int);
    int (* _Nullable _write)(void *, const char *, int);

    /* separate buffer for long sequences of ungetc() */
    struct __sbuf _ub; /* ungetc buffer */
    struct __sFILEX *_extra; /* additions to FILE to not break ABI */
    int _ur; /* saved _r when _r is counting ungetc data */

    /* tricks to meet minimum requirements even when malloc() fails */
    unsigned char _ubuf[3]; /* guarantee an ungetc() buffer */
    unsigned char _nbuf[1]; /* guarantee a getc() buffer */

    /* separate buffer for fgetln() when line crosses buffer boundary */
    struct __sbuf _lb; /* buffer for fgetln() */

    /* Unix stdio files get aligned to block boundaries on fseek() */
    int _blksize; /* stat.st_blksize (may be != _bf._size) */
    fpos_t _offset; /* current lseek offset (see WARNING) */
} FILE;
```

```
// Microsoft
struct _iobuf {
    char* _ptr;
    int _cnt;
    char* _base;
    int _flag;
    int _file;
    int _charbuf;
    int _bufsiz;
    char* _tmpfname;
};
typedef struct _iobuf FILE;
```

- ¿Cuáles son los punteros a funciones en la estructura?
  - Aplicar el conocimiento declaraciones complicadas para analizar el tipo de esos punteros a funciones
- ¿Por qué `fclose` no espera un `const FILE *`? Comparar con `strlen` y `strchar`

¿Consultas?

Fin de la clase