Compiladores

- Um compilador traduz um programa escrito numa linguagem A para um programa escrito numa linguagem B.

- Normalmente A é uma linguagem de alto nível (C, C++, Java, ...) e B é uma linguagem de baixo nível (assembly ou linguagem máquina).

- Nesta disciplina vamos aprender técnicas usadas na construção de compiladores

- Um compilador é um programa. Até agora vocês têm usado compiladores feitos por alguém (gcc, javac, ...)

- Ao final da disciplina poderão inventar a vossa própria linguagem e fazer um compilador para ela.

Panorâmica geral sobre as várias fases de um compilador.

- Análise Lexical
- Análise Sintática (Parsing)
- Análise Semântica

} Front End

- Optimização de código
- Geração de código

} Back End

O front end é independente da arquitectura do computador.

Análise Lexical

→ também designada por "scanning"

- O analisador lexical tem uma série de regras lexicais que definem aquilo que cada token pode ser.

- O analisador lexical também se encarrega de eliminar comentários e fazer substituições de macros (ex: #define em C), embora por vezes isso seja feito numa fase prévia de pré-processamento.

Exemplo em C:

```
int main () {
    int soma = 0;
    int i = 0;
    while ( i <= 10 ) {
        soma += i;
        i++;
    }
    return 0;
}
```

O analisador lexico vê isto como uma sequência de caracteres: 'i' 'n' 't' ' ' 'm' 'a' ...

Obtem como output uma sequência de tokens.

	token
int	keyword
main	keyword
(	lparen
)	rparen
{	l bracket
int	keyword
soma	ident
=	assign
0	number
;	semicolon
int	keyword
i	ident
=	assign
0	number
;	semicolon
while	keyword
(	lparen
i	ident
<=	relop
10	number
)	r paren
...	...

Em C:

```
if if while ( x ++ ( ) ;
```

↑

não tem erros lexicais

Tudo o que aparece são tokens válidos

```
if ( : $ 50 )
```

↑

erro lexical em C

$ não é um token válido

O mesmo em linguagem natural. Por exemplo, em português:

eu universidade à vou.

não tem erros lexicais. Todas as palavras existem num dicionário de português (i.e. fazem parte do lexico português.)

No entanto, tem erros gramaticais, tal como o if if while (... também tem erros gramaticais de acordo com as regras sintáticas da linguagem C.

eu vouuu à universidade

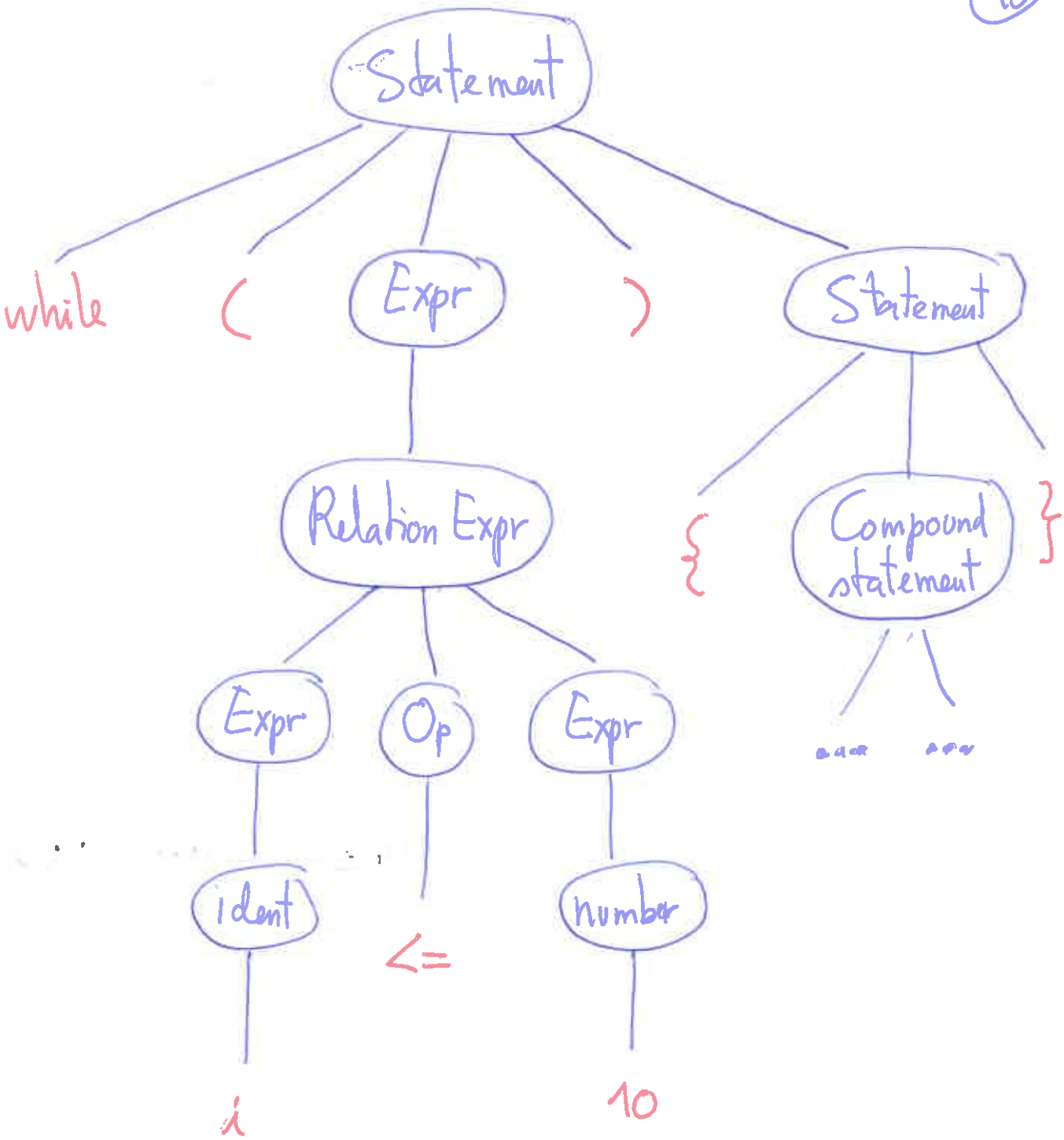
erro lexical

vouuu não faz parte do léxico português

Análise sintática

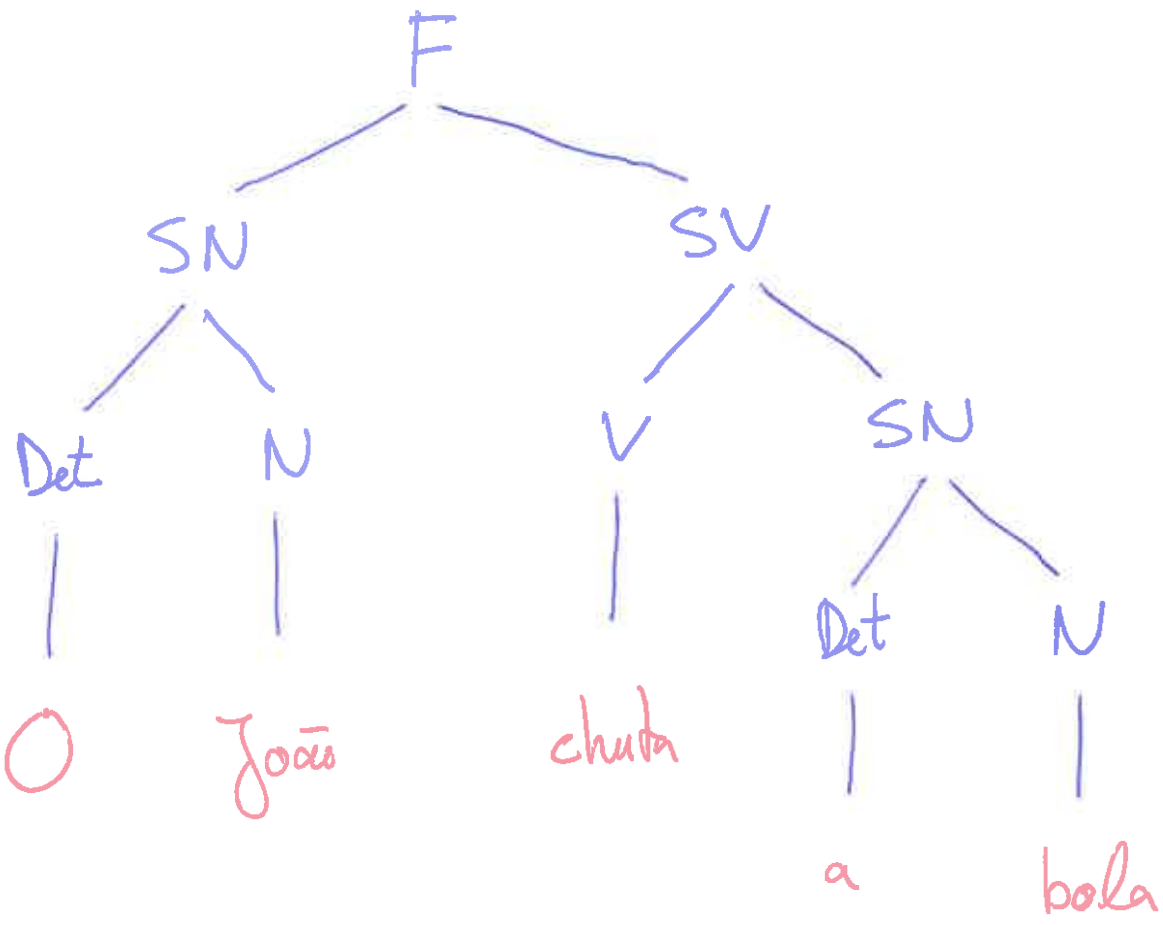
- Tenta impôr uma estrutura sobre a sequência de tokens.
- Para o fazer, necessita de saber as regras gramaticais da linguagem.
- A estrutura é uma árvore.
- Exemplo: só a parte do while no programa anterior

```
while ( i <= 10 ) {
    ...
}
```


Statement
while
(
Expr
)
Statement
Relation Expr
{
Compound statement
}
Expr
Op
Expr
ident
<=
number
i
10

O mesmo acontece em linguagem natural. Não basta juntar palavras do dicionário à balda para se construir frases.

As frases têm uma estrutura que é definida pelas regras sintáticas (ou gramaticais) da linguagem.



```
if if while ( x ++ ) ;
```

Tem erro sintático.

A sintaxe correcta para uma instrução if é

```
if ( Expr ) Statement
```

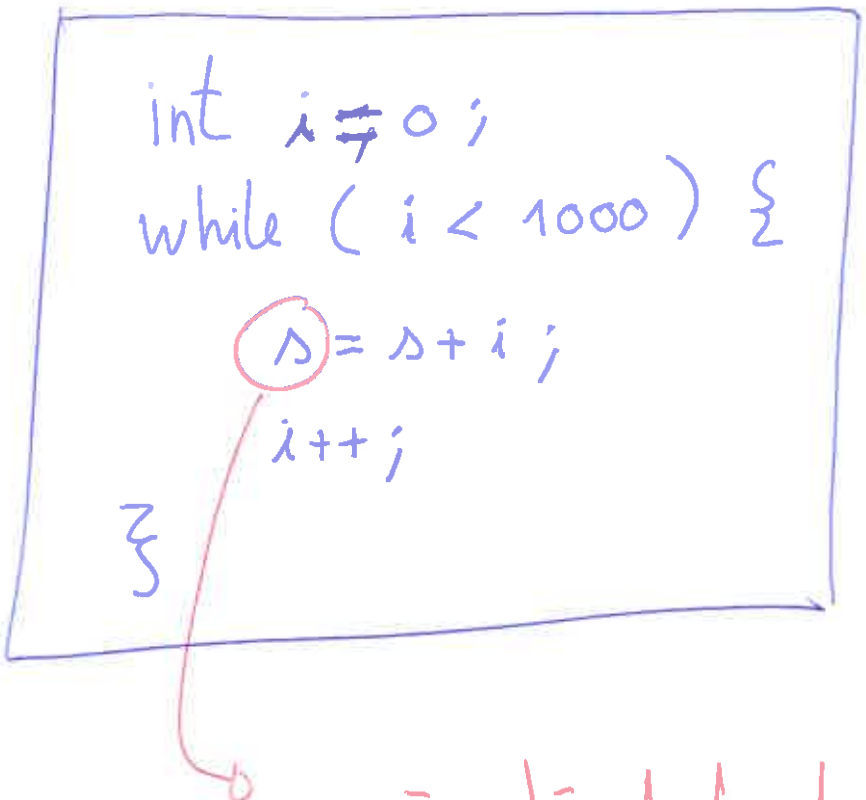
ou

```
if ( Expr ) Statement
else Statement
```

Análise semântica

- Verifica uma série de possíveis erros que não se consegue fazer apenas com a análise sintática.

- Exemplos: variáveis não declaradas.
  Verificação de tipos de dados.



Erro semântico

Optimização de código

→ não vamos falar nisto nesta disciplina.
(matéria avançada que podem aprender mais tarde. Vou apenas dar a ideia...)

Exemplo de optimização

```
int i=0, s=0;
while (i < 1000) {
    x = 7;
    s = s + i;
}
printf("%d %d", x, s);
```

$x = 7$ pode ser colocado fora do while.
Chama-se a isto code motion. Há muitas outras optimizações que um compilador pode fazer.

Geração de código

- Depende da máquina
- Gera código máquina directamente

  ou

  Gera assembly → código máquina