

COMPILADORES 2019/2020

aula 0x03 - (Mais) Sintaxe Abstracta

05/03/2020

Pedro Patinho <pp@di.uevora.pt>

Universidade de Évora - Departamento de Informática



1. Estruturas
2. Acções semânticas e produção de nós da APT
3. Geração de output a partir da APT
4. Implementação com Outras linguagens
5. Exercício

ESTRUTURAS

TIPOS DE DADOS PARA APT

```
1 struct calc_t_exp_ {
2     enum {EXP_NUM, EXP_ID, EXP_BINOP,
3         EXP_UNOP, EXP_PAR, EXP_ASSIGN} kind;
4
5     union {
6         int num;
7         char *id;
8         struct {
9             char op[3];
10            calc_t_exp arg1;
11            calc_t_exp arg2;
12        } binop;
13        . . .
14    } u;
15 };
```

TIPOS DE DADOS PARA APT

```
1 struct calc_t_seq_ {
2     enum {SEQ_EMPTY, SEQ_EXP} kind;
3
4     union {
5         struct {
6             calc_t_exp e;
7             calc_t_seq s;
8         } exp;
9     } u;
10 };
```

Outros tipos de nós (para linguagens completas):

- TYPE (int, float, string, void, ...)
 - para especificação de tipos
 - todas as expressões devem ter um nó destes
vai fazer falta para a análise semântica
- DECL (var, func, novo tipo - classe ou struct, ...)
- COND (if, if-then, if-then-else)
- LOOP (for, while, repeat-until, ...)
- STM (as 2 anteriores, return)

ACÇÕES SEMÂNTICAS E PRODUÇÃO DE NÓS DA APT

Regras passam a ter um tipo

```
1  %union {  
2      double val;  
3      char *name;  
4      calc_t_exp exp;  
5      calc_t_seq seq;  
6  }
```

```
1  %type    <exp>      exp  
2  %type    <seq>      seq
```


Literal Inteiro

```
1  exp: NUM { $$ = calc_exp_new_num($1); }
```

```
1  calc_t_exp calc_exp_new_num(int num)
2  {
3      calc_t_exp ret = (calc_t_exp) malloc (
          sizeof (*ret));
4
5      ret->kind = EXP_NUM;
6      ret->u.num = num;
7
8      return ret;
9  }
```

Afectação

```
1 | ID ASSIGN exp {$$=calc_exp_new_assign($1,  
    $3); }
```

```
1 calc_t_exp calc_exp_new_assign(char *id,  
    calc_t_exp rvalue) {  
2     calc_t_exp ret = (calc_t_exp) malloc (  
        sizeof (*ret));  
3     ret->kind = EXP_ASSIGN;  
4     ret->u.assign.id = id;  
5     ret->u.assign.rvalue = rvalue;  
6     return ret; }
```

Afectação

APT



GERAÇÃO DE OUTPUT A PARTIR DA APT

GERAÇÃO DE OUTPUT A PARTIR DA APT

```
1 void calc_exp_print(calc_t_exp exp)
2 {
3     switch (exp->kind) {
4     case EXP_NUM:
5         printf("[.exp [.num $%d$ ] ]\n",
6             exp->u.num);
7         break;
8     case EXP_ID:
9         printf("[.exp [.id $%s$ ] ]\n",
10             exp->u.id);
11         break;
12     . . .
```

ONDE CHAMAR A FUNÇÃO?

No símbolo inicial! (só executa a sua acção semântica no final do input)

ONDE CHAMAR A FUNÇÃO?

No símbolo inicial! (só executa a sua acção semântica no final do input)

```
1  input:  seq  { print_prologue();  
2              calc_seq_print($1);  
3              print_epilogue(); }  
4  ;
```

Aqui podemos chamar o verdadeiro “trabalho” do compilador!...

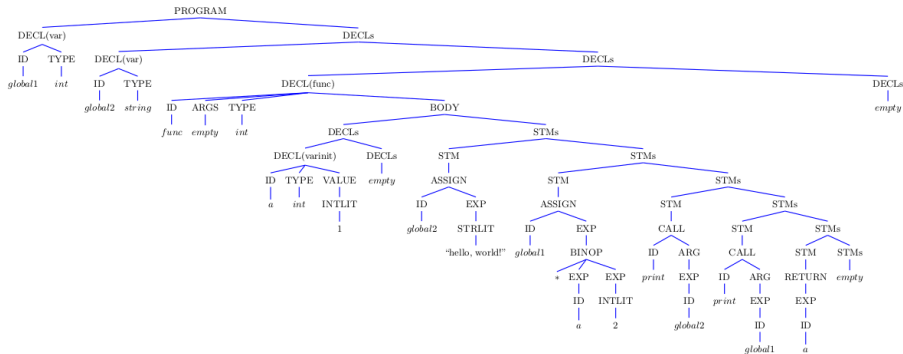
IMPLEMENTAÇÃO COM OUTRAS LINGUAGENS

OUTRAS LINGUAGENS (PARA ALÉM DO C)

- Classes (uma por regra)
- Construtores (um por produção)
- ou Classes para regras e Subclasses para produções
- ... e o resto é igual!

EXERCÍCIO

EXERCÍCIO



1. Escreva um programa que geraria a APT acima.