Parte 3

Principais dificuldades no desenvolvimento do compilador

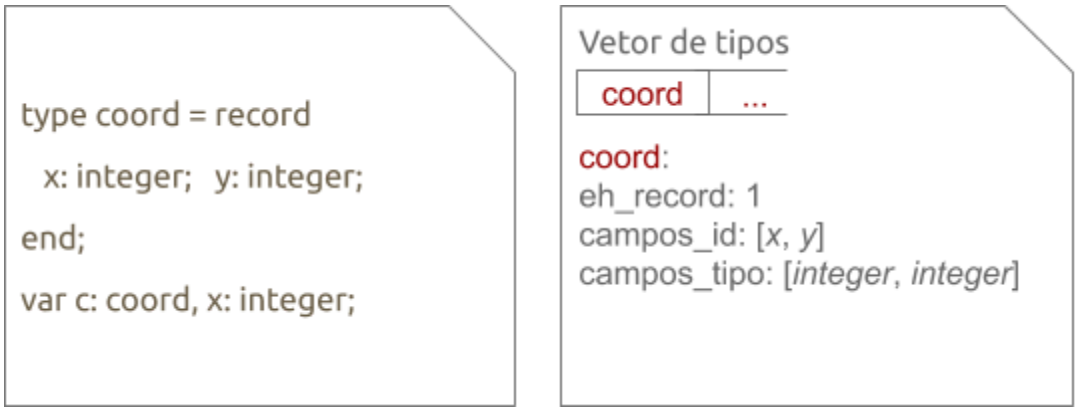
Os pontos que geraram maior dúvida e demandaram maior tempo para resolver:

- **Sinal negativo antes de expressão:** a solução foi empilhar a constante 0, depois empilhar o valor e realizar uma subtração para obter o valor negativo.
- **Parâmetros passados por referência:** fazer o uso correto de CRVL, CRVI e CREN foi um desafio.
- **Procedimentos/funções declarados dentro de outros procedimentos/funções:** foi preciso considerar que ao entrar em um procedimento é necessário desviar o fluxo do programa para os comandos desse procedimento para evitar que outros procedimentos declarados dentro deste sejam executados primeiro.
- **Funções usadas como parâmetros ao chamar outras funções:** foi necessário usar uma pilha para lidar com os parâmetros de cada função aninhada.

Inclusão do comando record

```
declara_record :==
        type <definiçao de record>
        <declaraçao de campos>; {<declaraçao de campos>;}
end;
definiçao de record :==
        <identificador> = record
declaraçao de campos :==
        <identificador> : <tipo>
```

Ao encontrar uma definição de record no código em pascal, o compilador deverá adicionar o nome do record ao vetor de tipos. Este é um vetor que será definida no código do compilador como um vetor de struct *tipo*, e *tipo* tem um campo *eh_record* que informa se o tipo declarado é um record e vetores *campos_id* para armazenar o identificador e *campos_tipo* para o tipo de cada campo filho do record.



Também é necessário incluir novos campos nos símbolos da tabela de símbolos: um que indica se o símbolo é um record (`eh_record`), quantos filhos tem (`qtd_campos`), um vetor que armazena os índices dos campos desse record (`vetor_de_campos`), um que indica se é campo filho de um record (`eh_campo`) e qual o índice na tabela de símbolos do record (`indice_record`) ao qual esse símbolo está atrelado. Ao encontrar uma variável com o tipo de um record, é preciso incluir um símbolo na tabela de símbolos para o record, e incluir símbolos para cada um dos campos que este tipo de record tem, com os nomes e tipos iguais aos definidos na definição record. Deve-se levar em consideração que é permitido criar variáveis com o mesmo nome do campo de um record, então símbolos com eh_campo em 1 devem ser ignorados ao declarar variáveis.

Nesse exemplo, serão incluídos três símbolos na tabela de símbolos: *c* (variável em si), *x* e *y*. É feito um AMEM para o número de campos do record (AMEM 2) e, após a declaração dos campos, o `vetor_de_campos` de *c* será atualizado. Na posição 0 e 1 ficam os endereços do primeiro e segundo campo: *x* e *y*. O `qtd_campos` é atualizado para 2 e `eh_record` recebe o valor 1. O `indice_record` de *x* e *y* recebe o endereço de *c*, o `eh_campo` recebe 1, os tipos podem ser encontrados procurando os tipos dos campos de coord no vetor de tipos, e o deslocamento é setado como se fossem adicionadas duas variáveis (var x, y : integer) ao invés de um record: se *c* for a primeira variável sendo declarada no programa principal (nível léxico 0), o nível léxico e deslocamento de *c.x* serão 0,0 e o de *c.y* serão 0,1.

Quando o campo de um record for acessado no código pascal, primeiro será necessário checar se esse campo existe para o record (usando o vetor de símbolos). Caso positivo, acessar o símbolo que tem este nome da tabela de símbolos e acessar a posição de memória normalmente (como é feito com variáveis de tipo integer, por exemplo). Para armazenar em *c.x*, usar ARMZ 0,0. O tipo do campo está definido no vetor de tipos.

Tabela de símbolos

c: eh_record = 1
vetor_campos [1, 2]

x: eh_campo = 1
indice_record = 0
deslocamento = 0
nivel = 0
tipo = integer

y: eh_campo = 1
indice_record = 0
deslocamento = 1
nivel = 0
tipo = integer