ALGOL 68 – FUNDAMENTOS DA LINGUAGEM

Aldo Ventura da Silva[1]

RESUMO

Na década de 60, as linguagens estruturadas imperativas de certa forma atendiam às necessidades da época. Com o tempo e a evolução tecnológica, foi necessário aumentar a velocidade e confiabilidade no desenvolvimento de software para atender às necessidades de negócio. Nessa evolução surgiu uma linguagem inovadora, com conceitos novos à época, que não foi bem explorada, mas que plantou a semente de inúmeros recursos que hoje são considerados básicos em qualquer linguagem de programação moderna. Essa linguagem é o Algol 68 e essa monografia apresentará alguns dos conceitos utilizados no projeto da linguagem que são relevantes nas linguagens atuais.

Palavras-chave: Algol, Linguagens de Programação.

[1] Bacharel em Sistemas de Informação – Universidade de São Paulo

1 INTRODUÇÃO

A linguagem Algol 68 foi introduzida na época em que as linguagens imperativas estruturadas eram as mais utilizadas, principalmente Fortran. Como evolução ao Algol 60, Algol 68 introduziu conceitos novos que à época não foram muito explorados principalmente devido a nenhum fabricante ter "patrocinado" a linguagem. Mesmo assim, muitos conceitos em linguagens posteriores foram baseados nos conceitos introduzidos pelo Algol 68.

Dentre esses conceitos, podemos citar os Tipos definidos pelos usuários, os arrays dinâmicos e o Polimorfismo, só pra citar alguns.

2 REFERENCIAL TEÓRICO

A principal referência para qualquer material referente à linguagem Algol 68 é o Revised Report (RR), que é o documento oficial de especificação da linguagem publicado em 1968. Porém o documento é conhecido por uma má reputação devido a sua dificuldade de leitura e complexidade para entender os conceitos apresentados.

A gramática usada no documento de especificação da linguagem não seguiu o padrão BNF (Backus-Naur), que era mais comum ao especificar comandos. Isso fez com que a linguagem fosse mais lenta para ser aceita.

Para esta monografia, a principal referência foi o documento publicado pelo autor Andrew S. Tanenbaum, que escreveu um tutorial de fácil entendimento sobre os principais conceitos sobre a linguagem.

3 ESTRUTURA BÁSICA DE UM PROGRAMA

No projeto da linguagem ALGOL 68, foi prevista a utilização de comandos compostos para delimitar as estruturas de controle. Para cada estrutura ou bloco, há um tipo de comando ou delimitador.

Um programa em ALGOL 68 consiste em um conjunto de expressões definidas e estruturas de controle delimitadas pelos comandos BEGIN e END.

As expressões, também conhecidas como unidades ou units, são separadas por ponto-e-vírgula e podem ser comandos, chamadas de rotinas e atribuições de variáveis, etc.

As estruturas de decisão e de repetição, também conhecidos como blocos, possuem comando de delimitação próprio.

Por exemplo, o IF possui como comando de término o FI, que nada mais é que o comando IF escrito ao contrário. A mesma regra vale para o comando CASE que possui como término o ESAC e o comando DO, que possui como término o OD, este último é usado para as estruturas de repetição FOR e WHILE.

Os comandos e expressões podem estar dispostos sem restrição para facilitar a leitura e entendimento dos programas, ou seja, pode-se utilizar de Tabulação e espaços livremente para alinhar as estruturas e identar os comandos de uma mesma estrutura, sem perigo de ocorrer problemas na compilação.

Os comentários também podem ser utilizados livremente. Há cinco comandos possíveis para essa funcionalidade. São eles: ¢, comment, co, # e £. Para todos eles, o comentário deve estar delimitado pelo comando escolhido.

Um programa em ALGOL 68 é consistido de pelo menos uma expressão válida entre os comandos BEGIN e END. O comando SKIP é um comando do tipo Dummy, e pode ser utilizado para validar estruturas semi-prontas, facilitando a compilação de programas ainda em construção.

Os comandos read e print, respectivamente, permitem manipular entrada e saída de dados.

Cadeias de caracteres, ou strings, são representadas como Aspas Duplas (“”).

3.1 Considerações e pontos relevantes

O conceito de delimitar o programa com BEGIN/END também é padrão na linguagem PASCAL, que usou Algol como inspiração.

A liberdade na codificação era um ponto interessante, pois não restringia ao preenchimento obrigatório de determinada coluna do código, como era comum em Fortran e Cobol. Isso fez com que fosse possível organizar as linhas de código de forma mais legível e de fácil entendimento, sem precisar obrigatoriamente preencher determinada coluna para que o compilador possa interpretá-la corretamente.

Essa liberdade de identação de código torna-se hoje uma boa prática de programação, tanto que em algumas ferramentas de desenvolvimentos atuais (Como Eclipse) são feitas automaticamente ao compilar os programas.

Exemplos dessa liberdade são os comentários, que podem ser feitos de várias formas. Isso mantem-se até hoje na maioria das linguagens, que definem comentários de linha e de bloco.

4 UNIDADES

Unidades ou Units em ALGOL 68 são os itens mais granulares para as expressões que constituem o contexto dos blocos e programas.

Essas unidades podem ser usadas em qualquer contexto que se espera expressões. Isso é possível devido a ortogonalidade, que é o principio ao qual a linguagem Algol 68 foi concebida.

Dentre as inúmeras variações de unidades, a seguir serão citadas as mais comuns.

4.1 Constantes

Valores fixos utilizados geralmente em inicialização de variáveis ou em fórmulas, geralmente representam os tipos mais primitivos de dados (int, real, char, bool, string).

Também é possível usar constantes em arrays ou em estruturas, desde que os valores sejam separados por parênteses. Abaixo alguns exemplos de constantes.

3 → constante do tipo int

5.2 → constante do tipo real

“y” → constante do tipo char

“ok” → constante do tipo string

((1,2,3), (8,9,10)) → constante do tipo array

4.2 Variáveis e Formulas

As variáveis são os identificadores criados pelo programador para representar dados na memória. As variáveis podem receber valores de vários tipos ou modos.

Uma variável pode receber valores constantes ou mesmo de outras variáveis ou da combinação de ambas através de fórmulas.

As fórmulas são a junção de um operador e um ou mais operandos.

Segue alguns exemplos:

Int x := 8 → declaração de uma variável denominada x que recebe o valor 8

real y:= x * 2 → variável do tipo real denominada y que recebe o valor da fórmula x * 2

4.3 Slices

Os slices são um recurso interessante em ALGOL 68 para trabalhar com trechos de arrays.

Essa funcionalidade permite atribuir/copiar valores de uma parte de um array para outro de forma que os endereços sejam válidos. Segue alguns exemplos:

a[3:8] := b[1:6] → os seis valores dos elementos da posição de 1 a 6 são copiados da variável array b para a variável array a na posição 3 a 8

matriz[5, 1:20] := b[1:20] → os 20 valores dos elementos da posição de 1 a 20 são copiados da variável array b para a variável matriz, porém na linha 5 e colunas de 1 a 20.

4.4 Seleções (Selections)

Seleção ou selection é o recurso em ALGOL 68 para atribuir ou recuperar um valor de um determinado campo em uma estrutura. Estruturas são tipos de dados definidos pelo programador para representar tipos abstratos de dados, que são formados pela combinação de uma ou mais variáveis.

Para acessar o atributo de uma estrutura, é necessário usar o comando of.

Segue abaixo um exemplo:

idade of rogerio := 18 → atribui o valor 18 ao campo idade da variável rogerio

4.5 Procedures

Procedures ou sub-rotinas são blocos que poder ser reutilizados inúmeras vezes, diminuindo a quantidade de código usado para se fazer uma determinada tarefa ou cálculo.

As procedures podem ter parâmetros, podem retornar valores e usados em atribuições.

4.6 Atribuições (Assignments)

Atribuições são basicamente a passagem de valor à variáveis previamente criadas. Ao efetuar a atribuição do valor à variável, o endereço de memória da variável passa a conter o valor. Quando a variável é utilizada posteriormente, o valor é recuperado no endereço da memória correspondente.

A atribuição se dá com a utilização do comando ":=", onde a variável a receber o valor fica do lado esquerdo e o valor no lado direito. Segue exemplo:

```
metade := valor / 2;
Quadrado := valor * valor;
```

4.7 Decisão

Cláusulas de decisão são expressões que avaliam determinada condição e executam expressões baseadas no resuldado da avaliação. Em ALGOL 68 há dois tipos. IF e CASE.

O IF possui a seguinte estrutura:

if <expressão logica>

then

<unidades se verdadeiro>

else

<unidades se falso>

fi

A expressão deve ter um retorno lógico, geralmente através de um operador de comparação (menor, igual, etc). As unidades a serem executadas quando a expressão é verdadeira ou falsa, pode conter uma ou mais expressões.

Essas unidades podem inclusive retornar um valor, caso a estrutura IF esteja sendo usada em uma expressão lógica.

Outra forma mais simples de utilizar a estrutura if é através de parênteses e o caracter pipe (|):

(<expressão logica> | <unidade se verdadeiro> | <unidade se falso>)

O CASE possui a seguinte estrutura:

case <expressão> in

expressão quando 1,

expressão quando 2,

expressão quando n,

out expressão padrão

esac

A expressão deve retornar um valor inteiro, que é avaliado e selecionado a expressão equivalente ao valor avaliado. Caso não exista a expressão de índice equivalente ao avaliado, a expressão OUT será executada.

4.8 Considerações e pontos relevantes

O conceito de unidades é o resultado do fundamento básico do projeto da linguagem Algol 68, que é a ortogonalidade. Esse conceito permite a construção de inúmeras expressões através de outras expressões menores ou cascateadas, cujo resultado pode se tornar a entrada para outra expressão, sem limite de aninhamento.

Um exemplo é a atribuição de valores usando-se estrutura de decisão. Segue um exemplo:

```
begin
Int i,j;
Read(i,j);
Int x = if i<j then i+j else j-i fi;
end
```

No exemplo acima, é possível ver claramente o conceito de ortogonalidade sendo aplicado na linha de atribuição da variável x, onde o bloco if é executado e seu retorno é utilizado na atribuição.

Outro ponto relevante é a identificação dos trechos dos arrays, que podem ser utilizados diretamente, sem precisar ir item a item da matriz. Esse recurso é um exemplo de como a estrutura da linguagem considera quase tudo como objetos distintos, com conversões implícitas na recuperação e atribuição dos valores. Esse recurso em si é pouco explorado atualmente, mas de certa forma é utilizado implicitamente na orientação à objeto, através de herança e compatibilidade de componentes.

Outro exemplo claro da ortogonalidade são as procedures, que são objetos individuais e podem ser utilizados inclusive em atribuições.

5 VARIÁVEIS E TIPOS DE DADOS (MODES)

As variáveis em ALGOL 68 devem obrigatoriamente ser declaradas. Em linguagens mais modernas, essa prática é recomendada, mas não obrigatória em algumas delas.

Uma característica interessante no ALGOL 68 é que as variáveis podem conter espaços. Por exemplo, a variável valortotal é igual a valor total, ou seja, ambas equivalem a um mesmo identificador.

A declaração deve ser feita informando o tipo ou modo seguido pelo nome da variável. É possível também informar mais variáveis para um mesmo modo, separando-as por vírgula.

Há basicamente quatro tipos primitivos de dados em ALGOL 68, também denominados Modos. São eles: int, real, bool e char.

Além dos tipos primitivos, a linguagem ALGOL 68 permite a criação de modos próprios ou utilizar variações dos modos já existentes.

Dentre outros tipos disponíveis, há também: string (Cadeia de caracteres), compl (número complexo), bits/bytes (palavra de máquina de Bits ou Bytes), sema (semáforo), format (usado para formatação de I/O) e file (Usado para I/O).

Além da declaração simples de variáveis, é possível usar modificadores na declaração permitindo maior controle ao programador, de acordo com suas necessidades.

Um exemplo é o modificador long ou short, que permitem declarar variáveis integer ou real com tamanho maior ou menor, respectivamente. Isso permite uma maior otimização da memória, onde o programador pode definir um tamanho mais coerente à sua utilização, caso o tamanho padrão não seja suficiente ou seja maior do que a real necessidade. Em linguagens mais modernas, esse recurso é feito de forma mais implícita.

No caso do modificador long e short, é possível combinar mais de uma ocorrência desse modificador, aumentando ou diminuindo ainda mais a capacidade da variável.

Outro modificador interessante é o ref, que é usado quando queremos declarar uma variável que na verdade é um ponteiro para um endereço de memória de determinado tipo. Com esse recurso, é possível trabalhar com referências à variáveis, abrindo novas

formas de trabalhar com elas. Uma das principais é a passagem de parâmetros por referência nas procedures, permitindo que os valores alterados dentro da procedure sejam refletidos nas variáveis das unidades chamadoras.

5.1 Considerações e pontos relevantes

Neste ponto já é possível notar o potencial do Algol 68 quando se diz respeito aos tipos de dados. A introdução dos modificadores faz com que um novo leque de opções se abre para os desenvolvedores.

Os tipos primitivos são simples, mas atendem a maioria das necessidades convencionais, mas os modificadores long e short permitem aumentar ou diminuir a capacidade de alguns tipos numéricos. Dessa forma o desenvolvedor pode otimizar a memória de acordo com o contexto da aplicação que ele irá criar. Nas linguagens modernas, esses tipos de dados derivados geralmente são nativos na linguagem, como se fosse tipos primitivos.

6 TIPOS DEFINIDOS PELO USUÁRIO

É possível declarar listas, ou arrays, de qualquer tipo primitivo, com qualquer dimensão. Também é possível utilizar variáveis na declaração dos arrays e não apenas constantes.

O modificador flex pode ser utilizado na declaração dos arrays, permitindo que a variável tenha seu tamanho alterado ao longo do programa. Um exemplo implícito desse recurso é o tipo string, que é um exemplo de um array unidimensional do tipo char que possui tamanho variado.

Também é possível utilizar variáveis que são ponteiros a um determinado tipo. Para isso, é necessário utilizar o modificador ref.

É possível definir tipos e estruturas próprias, onde o programador define um tipo composto, que consiste de dois ou mais tipos primitivos. Na declaração de variáveis de uma estrutura, é possível atribuir valores para cada um dos campos da estrutura,

semelhante à um Construtor de classes nas linguagens Orientadas à Objeto modernas.

Para referenciar os campos específicos de uma variável, é utilizado o comando of, da seguinte forma: campo of variável.

É possível declarar variáveis para receber mais de um tipo de dado, através do comando union. Ao tratar a variável, é possível verificar o tipo corrente da variável por uma estrutura Case.

Semelhante à definição de tipos próprios com o Struct, é possível definir tipos próprios com o comando mode. Isso facilita a declaração de variáveis de um mesmo tipo de estrutura. Com o mode também é possível definir estruturas do tipo lista ligada ou árvores recursivas através de ponteiro para o próprio tipo.

Também é possível definir procedures ou sub-rotinas através do comando proc, onde é definido parâmetros (opcionalmente) e o retorno da rotina (opcionalmente). Caso não exista retorno, deve ser utilizado o comando void.

6.1 Considerações e pontos relevantes

Aqui é onde o Algol 68 mostra sua principal inovação, que é a capacidade de trabalhar com tipos de dados definidos pelo desenvolvedor, derivando dos tipos mais primitivos.

A definição de estruturas e novos tipos permite que tipos de dados sejam criados especificamente para tratar as necessidades de negócio específicas da aplicação a ser criada. Dessa forma o desenvolvedor consegue aproximar os objetos de dados ao negócio, facilitando a leitura e entendimento do programa final. Isso foi de certa forma o passo inicial para a definição de Classes e seus atributos, conceito básico para a orientação à objetos atualmente.

Outra grande inovação foi a possibilidade de se trabalhar com arrays dinâmicos, ou seja, com tamanho variado. Isso faz com que os programas sejam sustentáveis, otimizando ainda mais o uso da memória.

Também é possível observar novamente o conceito de ortogonalidade ao possibilitar a utilização de variáveis na

declaração dos arrays. Em muitas linguagens atuais, isso não é possível. Nessa situação, a cada novo comando de declaração da variável array, a variável utilizada na definição do tamanho pode conter valores distintos, resultando em arrays de tamanhos distintos também.

A definição de ponteiros para tipos de dados primitivos ou não também é uma enorme contribuição, pois permite trabalhar com referências a tipos, além de que, se combinadas com uma estrutura ou um tipo definido pelo usuário, permite a utilização de listas dinâmicas. Dessa forma é possível definir uma biblioteca própria de tipos, para serem reaproveitadas em outras aplicações. Isso foi o embrião para a evolução para a orientação à objetos e as bibliotecas padrão das linguagens, algo comum atualmente, acelerando o desenvolvimento, pois o desenvolvedor usa as bibliotecas já prontas, sem se preocupar com bugs ou problemas já resolvidos por essas bibliotecas encapsuladas.

Em relação à orientação à objetos, outra grande inovação foi o recurso introduzido pelo comando union, que permite que uma variável permita receber vários tipos distintos. Isso se aplica não somente a variáveis, mas também às procedures, que podem ter retornos e parâmetros (assinaturas) distintos, mas com o mesmo nome. Este é o princípio do conceito de Polimorfismo, algo que é básico para qualquer linguagem orientada à objetos atualmente.

7 CONVERSÕES (COERCIONS)

A maioria das linguagens permite a conversão de tipos entre as variáveis. Em Algol 68, isso também é possível, na verdade, é até natural da linguagem, pois a conversão em si é tratada como uma unidade isolada.

Um exemplo de como essa conversão é tratada naturalmente em Algol 68 é o código abaixo.

```
Begin
Int i;
i:= 3;
```

```
Print(i);
end
```

No exemplo acima, mais especificamente na linha de atribuição da constante 3 à variávei i, há implicitamente uma conversão de tipos. O valor 3 é uma constante do tipo int, já a variávei i é um objeto do tipo ref int, ou seja, é um ponteiro da memória para um valor inteiro. Embora a atribuição pareça ser algo trivial e simples, há uma conversão interna devido à linguagem Algol 68 tratar tudo como se fossem objetos distintos.

As conversões em Algol são classificadas em basicamente 5 tipos. A seguir cada um dos tipos e uma breve explicação sobra a situação em que cada tipo é empregado.

7.1 Dereferencing

Esse é um tipo de conversão comum em exemplos como o citado acima, onde a entrada é um ponteiro para um determinado modo (ref m) e a saída é um valor desse determinado modo (m). Esse tipo de conversão acontece implicitamente na atribuição de variáveis e passagem de parâmetros.

7.2 Deproceduring

Esse é um tipo de conversão semelhante ao dereferencing, mas específico para retorno de chamadas de procedures, onde o resultado da procedure é de um determinado modo (proc m) e seu resultado será convertido para esse determinado modo (m).

7.3 Widening (alargamento)

Esse tipo de conversão é comum em operações aritméticas envolvendo tipos numéricos ou em cadeia de caracteres.

Um exemplo é quando é utilizado um valor inteiro para ser atribuído a uma variável do tipo real, embora sejam de tipos distintos (real e int), são compatíveis quando o tipo destino é real, portanto, o tipo int é “alargado” para o tipo real e a conversão pode

ser feita com sucesso. Nesse exemplo, o caminho inverso não é possível, pois o tipo real é maior que o tipo int e a conversão não pode sem a perda de valor/precisão.

Outro exemplo é a atribuição de um tipo char a uma variável do tipo string. O tipo string é na verdade um array de char ([] char), portanto, o tipo char é alargado para o tipo [] char.

7.4 Rowing

Esse tipo de conversão é usado quando há operações de atribuição envolvendo arrays, onde a conversão é feita atribuindo o tipo de dado (m) a uma fatia do array ([] m).

7.5 Uniting

Esse tipo de conversão é usado quando há envolvimento do modo UNION na atribuição de valores. Essa conversão é feita de forma implícita, resolvendo o tipo a ser utilizado e validando se a variável permite esse tipo.

7.6 Voiding

Esse tipo de conversão é usado quando não há conversão, por exemplo, uma procedure que não retorna valor. Caso ela seja declarada como sem retorno, se mesmo assim no corpo dela houver um retorno, ele será descartado, como se tivesse sido convertido para void.

7.7 Considerações e pontos relevantes

Este conceito de conversão nativo da linguagem é um grande avanço para a evolução à orientação a objetos, que é a amarração dinâmica. Essa conversão é fruto da ortogonalidade previsto no projeto da linguagem.

8 OPERADORES

A linguagem Algol 68 introduziu uma funcionalidade poderosa, que é a de se criar Operadores próprios, geralmente para operações envolvendo tipos de dados criados pelo usuário.

A definição de operadores facilita a manipulação dos dados, pois permite operação entre objetos de forma simples, sem a necessidade de procedures, facilitando a leitura do código.

8.1 Definindo novos operadores

O uso é semelhante ao de uma procedure, onde são necessário dois parâmetros, que seriam os operandos da expressão, e o caracter ou palavra que seria o operador.

Imagine um programa que tem um tipo de dado chamado vetor de 10 posições, que é um array de inteiros:

```
mode vetor = [1:10] int;
```

Caso seja necessário somar dois vetores e como resultado um terceiro vetor, uma das opções seria criar uma procedure que recebe dois vetores como parâmetro e devolve o vetor somado como resultado. Abaixo o código dessa procedure e a forma de utilização da mesma.

```
Proc somavetor = (vetor x,y) vetor:
Begin
                    vetor s;
                    for i to 10 do
                      s[i] := x[i] + y[i];
                    od
end;


                    vetor v1, v2, v3, vr;
                    read ((v1, v2, v3));
                    vr := somavetor(somavetor(v1, v2), v3);
```

O exemplo acima, embora funcione perfeitamente, poderia ser escrito de uma forma diferente, utilizando-se do recurso de definir um novo operador, facilitando a utilização do mesmo, evitando o uso de procedure aninhada.

```
op +vet = (vetor x,y) vetor:
Begin
                    vetor s;
                    for i to 10 do
                      s[i] := x[i] + y[i];
                    od
end;

                    vetor v1, v2, v3, vr;
                    read ((v1, v2, v3));
                    vr := v1 +vet v2 +vet v3;
```

Com o uso de operadores, a utilização dos mesmos facilita a leitura e entendimento, principalmente quando há chamadas aninhadas, como no exemplo anterior.

Nesse caso, o operador foi definido como +vet, porém ele poderia ser simplesmente o caracter +. Isso é possível pois o Algol 68 sabe diferenciar o operador a ser utilizado de acordo com o contexto no qual ele se encontra.

O exemplo abaixo mostra de forma clara como esse determinação do operador funciona na linguagem Algol 68.

```
Begin
Op ? = (int i, j) real: i+j;
Op ? = (int I, real z) real: i-z;
Op ? = (real z, int i) real: i+z+19;
Op ? = (real z, y) real (x<y | z | y);
Print (2?9, 6?2.0, 3.14?8, 9.2?9.9);
end
```

O exemplo acima mostra o operador ? sendo definido de várias maneiras diferentes, de acordo com os parâmetros no contexto de sua utilização. A saída para esse programa é a seguinte: 11.0, 4.0, 30.14, 9.2.

Essa avaliação sobre qual “rotina” utilizar é semelhante a quando usamos o comando union para definir uma variável que pode receber vários tipos. Essa avaliação acontece naturalmente quando se usa o operador de soma (+) em valores do tipo int e real. Embora seja o mesmo operador, o compilador precisa direcionar comandos distintos ao hardware conforme o tipo de dado. Por isso o comportamento em avaliar diferentes tipos e rotinas para um mesmo operador é natural para a linguagem Algol 68.

8.2 Definindo prioridades entre os operadores

Além de permitir a definição de operadores específicos, a linguagem Algol 68 permite definir a prioridade entre os operadores.

Essa prioridade também é conhecida como ordem de precedência. A ordem de uma fórmula pode ser definida usando-se parênteses. Quando não há parênteses, a ordem de precedência nativa da linguagem é utilizada.

Por exemplo, no comando print(6+3*5), o resultado esperado é 21, pois embora não exista parênteses, é comum resolver primeiro a multiplicação para depois resolver a soma.

Essa prioridade da multiplicação ao invés da soma pode ser alterada conforme necessidade do usuário. Segue um exemplo de redefinição da ordem desses operadores:

```
Begin
        prio + = 3, * = 2;
        print (6+3*5);
end
```

Após a redefinição da prioridade, o programa acima exibe como resultado o valor 45.

8.3 Considerações e pontos relevantes

O conceito de redefinir operadores ou definir operadores próprios é uma inovação usada atualmente também em orientação à

objetos. Nem todas as linguagens permitem, mas as que permitem chamam isso e Sobrecarga de Operadores (Operator Overload).

A redefinição de prioridades é pouco explorada atualmente, mas já era possível ver o quão versátil era a linguagem Algol 68.

CONCLUSÃO

Conforme pode ser visto, a linguagem Algol 68 possui muitas semelhanças às linguagens atuais, não só pela sintaxe e estrutura básica, mas também pelos recursos inovadores por ela introduzidos.

Dentre os conceitos introduzidos, destacam-se os tipos de dados e estruturas que podem ser definidos pelo usuário, permitindo uma extensão aos tipos primitivos nativos da linguagem. Também vale destacar a possibilidade de arrays dinâmicos e a introdução do conceito de polimorfismo através do comando union.

Embora a linguagem Algol 68 não teve tanto sucesso como outras linguagens mais comerciais, ela é referência de muitas outras linguagens, que se inspiraram nos conceitos introduzidos por ela e evoluíram suas funcionalidades até os dias atuais, onde podemos trabalhar de forma natural com conceitos de orientação à objetos, que na época do Algol ainda nem se falava, mas a linguagem previa uma generalização semelhante.

Embora a linguagem não teve um patrocínio de nenhum fabricante para se tornar popular, ela é uma protagonista na história das linguagens por abrir caminho para uma série de linguagens inspiradas em seus conceitos que se tornaram mais populares. Alguns exemplos de “descendentes” do Algol são: Pascal, C, C++, Simula, Smalltalk.

Das linguagens mais comum usadas atualmente, como Java e C#, ambas derivadas do C++, todas tem eu suas raízes conceitos introduzidos pelo Algol.

REFERÊNCIAS

Koster, Kees; “The Making of Algol 68”; C.H.A. Koster; www.algol68.org

Tanenbaum, Andrew S.; “A tutorial on Algol 68”; ACM Computing Surveys, Vol. 8, No. 2