

CENTRO UNIVERSITÁRIO FEI

Andrias Matheus
Gabriel Scopel
Fernando Shiraishi
Rafael Machado

COMPILADOR BAR

Compiladores - Charles Ferreira

São Bernardo do Campo
2024

ÍNDICE

[COMPILADOR BAR](#)

[ÍNDICE](#)

[INTRODUÇÃO](#)

[TOKENS](#)

[GRAMÁTICA](#)

[VARIÁVEIS](#)

[FUNÇÕES](#)

[EXPRESSÕES](#)

[FLUXO](#)

[CONTROLE](#)

[EXECUÇÃO](#)

[EXEMPLO](#)

INTRODUÇÃO

Este projeto tem como objetivo a criação de um Compilador utilizando técnicas aprendidas durante as aulas da disciplina de Compiladores. O compilador deve ser capaz de realizar a conversão de um programa desenvolvido na Linguagem definida pelo grupo para a Linguagem C.

O compilador deverá receber como entrada um arquivo contendo um programa escrito na Linguagem definida pelo grupo e gerar uma forma equivalente em C.

A linguagem criada pelo grupo foi batizada de **B.A.R - Boteco, Álcool e Risada**, e utiliza objetos e acontecimentos comuns em bares como comando de código.

TOKENS

COMENTARIO -> // (ID | NUMBER | CONJUNCAO | DELIMITADORES | OPERADORES) +

ID -> [a-z]+ | [A-Z]+

NUMBER -> NUM | FLUTUANTE

FLUTUANTE -> NUM . NUM

NUM -> [0-9]+

CONJUNCAO -> & | && | | ||

DELIMITADORES -> (|) | { | } | [|] | = | == | , | ; | : | ' | "

OPERADORES -> + | - | * | / | % | ** | < | <= | > | >= | ! | !=

TYPES -> dose | latinha | truco | shot

RESERVADAS -> amendoim | torresmo | velho | barreiro | velhoBarreiro | bebe | canta | PT

GRAMATICA

VARIÁVEIS

```

declara -> ( truco ID codigo ) | ( tipoDado ID ( atribuicao2 | ; codigo ) )
atribuicao -> ID atribuicao2
atribuicao2 -> = ( expressaoAritimetica | string ) endCode

```

FUNCOES

```
print -> canta ( ( id | string ) ) endCode
scan -> bebe ( argumentoScan ) endCode
argumentoScan -> “ % tipoScan “ ( eps | & ) ID
tipoScan -> d | s | f | lf
```

EXPRESSOES

```
expressaoAritmetica -> fator expressaoAritmetica2
expressaoAritmetica2 -> ε | operadorMatematico fator expressaoAritmetica
                        ( endCode | expressaoAritmetica )
```

fator -> numero | ID | (expressaoAritimetica)

FLUXO

```
fluxo -> ( codigo_if | codigo_while | codigo_for )
codigo_if -> velho ( condicao ) { codigo } codigo_elif codigo_else codigo
codigo_elif -> ε | velhoBarreiro ( condicao ) { codigo } codigo_elif codigo
codigo_else -> ε | barreiro { codigo } codigo
codigo_while -> amendoim ( condicao ) { codigo } codigo
codigo_for -> torresmo ( ID; condicao; incremento ) { codigo } codigo
```

```
condicao -> termoCondicao operador termoCondicao
termoCondicao -> " operadorMatematico " | fator | expressaoAritimetica
incremento -> ID ( somaSubtracao somaSubtracao | = expressaoAritimetica )
somaSubtracao -> + | -
```

CONTROLE

endCode -> ; codigo | PT

```
numero -> NUMBER
string -> " ( ID | operadorMatematico ) * "
tipoDado -> TYPES
```

```
operadorMatematico -> - | + | * | / | % | **
operadorLogico     -> == | != | < | > | <= | >=
```

EXECUÇÃO

Criamos um script para fazer a execução do compilador de forma mais automatizada, ele chama `compiler.sh`. Esse script é responsável por compilar e executar o código em java do compilador e idetar, compilar e executar o código de saída gerado em C.

A indentação do código de saída é feita utilizando o pacote *astyle*, e caso não seja encontrado no sistema em que o script está sendo executado, ele será instalado automaticamente.

Para executar o script, basta seguir os seguintes comandos a partir da pasta raiz do projeto:

Conceder permissões para o script do compilador: `chmod u+x ./compiler.sh`

Executar o script: `./compiler.sh`

OBS: O script `.sh` só roda de forma totalmente automática no Linux. Para rodar no Windows, basta executar pelo aplicativo do Git Bash, porém, é necessário baixar o *astyle* antes.

EXEMPLO

```
latinha a;
latinha b;
latinha c;
dose d;

canta("Programa Teste");
canta("Digite A");
bebe("%d", &a);
canta("Digite B");
bebe("%d", &b);

velho(a < b){
    c = a + b;
}
barreiro{
    c = a - b;
}

canta("C e igual a");
canta("%d", c);

d = c / (a + b);

canta("D e igual a");
canta("%f", d);
```

```
#include <stdio.h>

int main() {
    int a ;
    int b ;
    int c ;
    float d ;
    printf ( "Programa Teste " ) ;
    printf ( "Digite A " ) ;
    scanf ( "%d", &a ) ;
    printf ( "Digite B " ) ;
    scanf ( "%d", &b ) ;
    if ( a < b ) {
        c = a + b ;
    }
    else {
        c = a - b ;
    }
    printf ( "C e igual a " ) ;
    printf ( "%d", c ) ;
    d = c / ( a + b ) ;
    printf ( "D e igual a " ) ;
    printf ( "%f", d ) ;
    return 0;
}
```

```
latinha numero;
latinha primo = 1;

canta("Digite um numero");
bebe("%d", &numero);

velho (numero <= 1){
    primo = 0;
}
barreiro {
    latinha i = 2;
    boateAzul (i; i <= (numero / 2); i++){
        velho ((numero % i) == 0){
            primo = 0;
            saidera;
        }
    }
}

velho (primo == 1){
    canta("0 numero e primo");
}
barreiro{
    canta("0 numero nao e primo");
}
```

```
#include <stdio.h>

int main() {
    int numero ;
    int primo = 1 ;
    printf ( "Digite um numero " ) ;
    scanf ( "%d", &numero ) ;
    if ( numero <= 1 ) {
        primo = 0 ;
    }
    else {
        int i = 2 ;
        for ( i ; i <= ( numero / 2 ) ; i ++ ) {
            if ( ( numero % i ) == 0 ) {
                primo = 0 ;
                break ;
            }
        }
    }
    if ( primo == 1 ) {
        printf ( "0 numero e primo " ) ;
    }
    else {
        printf ( "0 numero nao e primo " ) ;
    }
    return 0;
}
```