

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA**

ALEXANDRE DOS SANTOS BINDO

**CAROLINE JENUARIO RODRIGUES
DA SILVA**

**MELQUISEDEQUE CASAGRANDE ALVES MENDES
DIAS**

RICARDO BERTOLIN

PROJETO COMPILADOR

CURITIBA

2023

ALEXANDRE DOS SANTOS BINDO

**CAROLINE JENUARIO RODRIGUES
DA SILVA**

**MELQUISEDEQUE CASAGRANDE ALVES MENDES
DIAS**

RICARDO BERTOLIN

PROJETO COMPILADOR

Projeto de Linguagens Formais e Compiladores da Graduação
em Engenharia da Computação da Pontifícia Universidade
Católica do Paraná, conferindo a primeira etapa.

Prof. Frank Coelho de Alcantara

CURITIBA

2023

SUMÁRIO

1 COMPILADOR LFC..... 4

1.1 CONTEXTO..... 4

1.2 DEFINIÇÃO DE LINGUAGEM 4

1.3 RESTRIÇÕES DE LINGUAGEM..... 5

1.4 EXEMPLOS DE APLICAÇÃO..... 6

1.5 RESULTADO DA ETAPA^[OBJ]..... 8

REFERÊNCIAS..... 8

1 PROJETO COMPILADOR

1.1 CONTEXTO

O trabalho em questão é destinado ao desenvolvimento de um compilador em 3 etapas, envolvendo uma nova linguagem de programação. Será baseado no microcontrolador ATmega2560, utilizando ferramentas secundárias de softwares para demais etapas, como GitHub, Repl.it, sendo as demais definidas posteriormente, incluindo a parte de hardware, qual foi optado pelo Arduino Mega para execução.

1.2 DEFINIÇÃO DE LINGUAGEM

O código utilizará uma linguagem com a gramática definida abaixo, qual consiste em declarações definidas, considerando seu conjunto (qual contém), incluindo a definição para o lexema **basic**:

<i>PROGRAM</i>	<i>BLOCK</i>
<i>BLOCK</i>	<i>{DECLS STMTS}</i>
<i>DECLS</i>	<i>DECLS DECL </i>
<i>DECL</i>	<i>TYPE ID ;</i>
<i>TYPE</i>	<i>TYPE [NUM] BASIC</i>
<i>STMTS</i>	<i>STMTS STMT E</i>

Lexema **basic**, expressando tipos básicos:

<i>STMT</i>	<i>IF (BOOL) STMT</i>
	<i>IF (BOO) STMT ELSE STMT</i>
	<i>WHILE (BOOL) STMT</i>
	<i>DO STMT WHILE (BOOL);</i>
	<i>BREAK ;</i>

Considerando regras de precedência, será utilizado o não-terminal **factor** para lidar com expressões.

<i>BOOL</i>	<i>BOOL JOIN JOIN</i>
<i>JOIN</i>	<i>JOIN && EQUALITY EQUALITY</i>
<i>EQUALITY</i>	<i>EQUALITY == REL EQUALITY != REL REL</i>
<i>REL</i>	<i>EXPR < EXPR EXPR <= EXPR EXPR >= EXPR EXPR > EXPR EXPR</i>
<i>EXPR</i>	<i>EXPR + TERM EXPR - TERM TERM</i>
<i>TERM</i>	<i>TERM * UNARY TERM / UNARY UNARY</i>
<i>UNARY</i>	<i>UNARY - UNARY FACTOR</i>
<i>FACTOR</i>	<i>(BOOL) NUM REAL TRUE FALSE</i>

Os lexemas **num** e **real** são usados para representar números inteiros e números de ponto flutuante de 16 bits, conforme serão necessários em futuras etapas.

<i>STMT</i>	<i>TYPE ID = NUM REAL TRUE FALSE</i>
	<i>DIGITAL_READ (PIN)</i>
	<i>DIGITAL_WRITE (PIN, NUM)</i>
	<i>ANALOG_READ (PIN)</i>
	<i>ANALOG_WRITE (PIN, NUM)</i>
	<i>PIN_MODE (NUM, PIN_TYPE)</i>
	<i>DELAY (NUM)</i>

1.3 RESTRIÇÕES DA LINGUAGEM

A linguagem elaborada não irá compreender estruturas de maior complexidade, como: Falta de Suporte a Arrays Multidimensionais, falta de estruturas de controle Avançadas (for, switch, foreach): características de recursividade, funções definidas.

O documento em questão não aborda, apesar da gramática estar bem definida, a forma como ela é usada no código, necessitando implementar um analisador sintático (em anexo) que segue essa gramática para verificar a validade dos programas. Também não é abordado diretamente nesta etapa, como a linguagem permite interação direta com hardware, além de como a linguagem lida com erros ou exceções.

1.4 EXEMPLOS DE APLICAÇÃO

Os seguintes exemplos básicos de aplicação para o Arduino Mega, demonstrando sua funcionalidade e aplicação.

Piscar um LED:

```
int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH); // Liga o LED
  delay(1000); // Aguarda 1 segundo
  digitalWrite(ledPin, LOW); // Desliga o LED
  delay(1000); // Aguarda 1 segundo
}
```


Ler um Sensor de Temperatura (DHT22):

```
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  float temperatura = dht.readTemperature(); // Lê a temperatura em Celsius
  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.println(" °C");
  delay(2000); // Aguarda 2 segundos entre as leituras
}
```

Controlar um Servo Motor:

```
#include <Servo.h>
```

```
Servo myservo;
```

```
void setup() {  
  myservo.attach(9); // Anexa o servo ao pino 9  
}
```

```
void loop() {  
  for (int pos = 0; pos <= 180; pos += 1) { // Move o servo de 0 a 180 graus  
    myservo.write(pos);  
    delay(15);  
  }  
  delay(1000); // Aguarda 1 segundo  
  for (int pos = 180; pos >= 0; pos -= 1) { // Move o servo de 180 a 0 graus  
    myservo.write(pos);  
    delay(15);  
  }  
  delay(1000); // Aguarda 1 segundo  
}
```

1.5 RESULTADO DA ETAPA

O presente documento descreve a entrega da primeira fase, qual será composta da apresentação da linguagem definida, destacando as regras de produção criadas para interação com o Hardware. Além disso, em anexo, será incluso exemplos de código utilizando a sua linguagem e que apresentem todas as funcionalidades, assim como a segunda parte da primeira entrega consiste na criação de um analisador léxico, capaz de validar todos os possíveis lexemas da linguagem, usando máquinas de estado finito, implementadas em Python, qual para teste, inclui arquivos de texto que contenham todos os lexemas da linguagem, com saída sendo um string. Finalmente, procura-se garantir que a linguagem está corretamente definida atende todos os requisitos especificados, também, o analisador léxico analisar a linguagem indicando os erros existentes.

REFERÊNCIAS

AHO, Alfred. LAM. S. Monica. SETHI. Ravi. ULLMAN, D. Jeffrey
Compiladores: Princípios, Técnicas e Ferramentas. 30 de Outubro de 2007.

Documentação do Arduino Mega.<https://docs.arduino.cc/hardware/mega-2560>.

<https://frankalcantara.com/construcao-interpretadores/>