



PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA
ENGENHARIA DE COMPUTAÇÃO

PROJETO COMPILADOR
Fase 1

CURITIBA
2023

CARLOS EDUARDO MARQUES ASSUNÇÃO TORRES
MILENA HELOÍSA DE AMORIM SILVÉRIO
RICARDO GODOI KURASHIKI

PROJETO COMPILADOR

Fase 1

Relatório apresentado à disciplina de Linguagem Formais e Compiladores, 9º Período do curso de Engenharia de Computação da Escola Politécnica da PUCPR. Orientado pelo professor Frank Coelho de Alcantara.

CURITIBA

2023

Sumário

1. DESCRIÇÃO DA FASE 1	4
2. DETALHAMENTO DA LINGUAGEM DESENVOLVIDA	6
3. CÓDIGOS	8
3.1. Código I	Erro! Indicador não definido.
3.2. Código II	Erro! Indicador não definido.
3.3. Código III	Erro! Indicador não definido.

1. DESCRIÇÃO DA FASE 1

O presente estudo visa à elaboração de uma gramática formalizada, juntamente com suas regras de produção, com o propósito de criar uma linguagem de programação. Essa linguagem será desenvolvida nas próximas etapas do projeto e será implementada em um microcomputador selecionado pela equipe.

A gramática, que será detalhada posteriormente, foi inicialmente concebida com base no esboço apresentado a seguir:

- Declarações:

Figura 1. Declarações iniciais

<i>program</i>	→	<i>block</i>
<i>block</i>	→	{ <i>decls stmts</i> }
<i>decls</i>	→	<i>decls decl</i> ϵ
<i>decl</i>	→	<i>type id</i> ;
<i>type</i>	→	<i>type</i> [num] basic
<i>stmts</i>	→	<i>stmts stmt</i> ϵ
<i>stmt</i>	→	<i>loc = bool</i> ; if (<i>bool</i>) <i>stmt</i> if (<i>bool</i>) <i>stmt</i> else <i>stmt</i> while (<i>bool</i>) <i>stmt</i> do <i>stmt</i> while (<i>bool</i>) ; break ; <i>block</i>
<i>loc</i>	→	<i>loc</i> [<i>bool</i>] id

Fonte: Frank Alcantara, 2023

- Expressões de produção:

Figura 2. Regras de produção iniciais

<i>bool</i>	→	<i>bool</i> <i>join</i> <i>join</i>
<i>join</i>	→	<i>join</i> && <i>equality</i> <i>equality</i>
<i>equality</i>	→	<i>equality</i> == <i>rel</i> <i>equality</i> != <i>rel</i> <i>rel</i>
<i>rel</i>	→	<i>expr</i> < <i>expr</i> <i>expr</i> <= <i>expr</i> <i>expr</i> >= <i>expr</i> <i>expr</i> > <i>expr</i> <i>expr</i>
<i>expr</i>	→	<i>expr</i> + <i>term</i> <i>expr</i> - <i>term</i> <i>term</i>
<i>term</i>	→	<i>term</i> * <i>unary</i> <i>term</i> / <i>unary</i> <i>unary</i>
<i>unary</i>	→	! <i>unary</i> - <i>unary</i> <i>factor</i>
<i>factor</i>	→	(<i>bool</i>) <i>loc</i> num real true false

Fonte: Frank Alcantara, 2023

Além do esboço fornecido pelo professor, será necessário adicionar novos itens à gramática, novas regras de produção e novas funções de interação com o *hardware*. Dentre as principais funções de interação com o *hardware*, pode-se citar:

- Ler e escrever em pinos digitais;
- Ler e escrever nas portas seriais;
- Ler e escrever em componentes opcionais como conversores digitais – analógico e analógico – digital.

A entrega desta fase será composta da apresentação da linguagem definida, destacando os novos itens da gramática, as regras de produção criadas e as funções de interação com o *hardware*. Além da apresentação dos itens anteriores, também é necessário a criação de no mínimo três exemplos de códigos, utilizando todas as funcionalidades definidas para a linguagem.

2. DETALHAMENTO DA LINGUAGEM DESENVOLVIDA

Conforme explicitado anteriormente, segue abaixo a gramática reestruturada com os novos itens adicionados.

Figura 3. Gramáticas da linguagem

GRAMÁTICA	
program	block
block	{ decls stmts ret }
decls	decls decl
	decls func_decl
	vazio
decl	type id;
type	type[num]
	basic
func_decl	basic id(args) block
args	arg_list
	vazio
arg_list	arg_list, type id
	type id
stmts	stmts stmt
	vazio
stmt	loc = bool;
	loc = hw_interact;
	loc = id(args);
	hw_interact
	id(args);
	for (loc = int; equality; loc = expr) stmt
	ctrl_struct
	while (bool) stmt
	do stmt while (bool)
	break;
	block
loc	loc[bool]
	id
ctrl_struct	if (bool) stmt ctrl_struct
	elseif (bool) stmt ctrl_struct
	else stmt
	vazio
hw_interact	pinMode int, int
	digitalRead int
	digitalWrite int, loc
	digitalWrite int, boolean
	analogRead int
	analogWrite int, loc
	analogWrite int, int
	serialBaud int
	serialAvailable
	serialRead
	serialWrite loc
	serialWrite int
ret	return rel;
	vazio

Fonte: Os autores, 2023

Além da gramática desenvolvida, também foram feitas alterações nas regras de produção, como pode ser observado abaixo.

Figura 4. Regras de produção da linguagem

REGRAS DE PRODUÇÃO	
bool	bool join
	join
join	join && equality
	equality
equality	equality == rel
	equality != rel
	rel
rel	expr < expr
	expr <= expr
	expr > expr
	expr >= expr
	expr
expr	expr + term
	expr - term
	term
term	term * unary
	term / unary
	unary
unary	-unary
	!unary
	factor
factor	(bool)
	loc
	int
	float
	boolean
boolean	true
	false

Fonte: Os autores, 2023

Por fim, os tipos básicos que serão utilizados na linguagem são detalhados abaixo.

Figura 5. Tipos básicos de dados

basic
int
float
void
boolean


Fonte: Os autores, 2023

3. CÓDIGOS

Os códigos em seguida foram desenvolvidos com base na gramática e regras criadas e especificadas anteriormente. Os códigos de exemplo buscam utilizar todas as funcionalidades disponíveis na língua.

3.1. Código I

Figura 6. Primeiro exemplo de aplicação da linguagem



```
{
  int x;
  int sensorValue;
  float y[3];
  boolean flag;
  void printSum(int a, int b)
  {
    int sum;
    int doSum(int a, int b)
    {
      return a + b;
    }

    sum = doSum;
    serialWrite sum;
  }

  serialBaud 9600;

  for (x = 0; x < 3; x = x + 1)
    y[x] = x * 2/1.5;

  printSum(5, 3);

  if (y[0] > y[1])
    flag = true;
  elseif (y[0] < y[1])
    flag = false;
  else
    flag = true;

  pinMode 10, 3;
  digitalWrite 10, flag;

  x = 10;
  while (x > 0)
  {
    x = x - 1;

    if (x == 5)
      break;
  }

  pinMode 15, 1;
  sensorValue = analogRead 15;
  serialWrite sensorValue;
}
```

Fonte: Os autores, 2023

3.2. Código II

Figura 7. Segundo exemplo de aplicação da linguagem

```
{
  int sensorValue;
  boolean available;
  int readValue;
  int calculateSum(boolean flag)
  {
    int sum;
    int j;

    sum = 0;
    for (j = 1; j <= 10; j = j + 1)
    {
      if (flag)
      {
        sum = sum + j;
      }
      else
      {
        sum = sum - j;
      }
    }

    return sum;
  }

  serialBaud 9600;

  pinMode 15, 1;
  sensorValue = analogRead 15, i;

  pinMode 12, 3;
  if (sensorValue > 100)
  {
    digitalWrite 12, true;
  }
  else
  {
    digitalWrite 12, false);
  }

  while (true)
  {
    available = serialAvailable;


    if (available)
    {
      readValue = serialRead;
      break;
    }
  }

  serialWrite readValue;
}
```

Fonte: Os autores, 2023

3.3. Código III

Figura 8. Terceiro exemplo de aplicação da linguagem



```
{
  int sensorValue;
  int serialValue;
  boolean available;
  boolean isDayTime;

  pinMode 14, 1;
  sensorValue = analogRead 14;

  if (sensorValue >= 100)
    isDaytime = true;
  else
    isDaytime = false;

  serialBaud 9600;

  do
  {
    available = serialAvailable;
  } while (!available)

  serialValue = serialRead;

  pinMode 13, 3;
  if (serialValue == 49)
  {
    digitalWrite 13, true;
  }
  else
  {
    digitalWrite 13, false;
  }
}
```

Fonte: Os autores, 2023