Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
COMPILADORES
Professora: Mariza Andrade da Silva Bigonha
TRABALHO PRÁTICO III
10 PONTOS
13/11/2017

1 Informações Gerais

A sua tarefa é construir um **Tradutor** da linguagem intermediária gerada pelo *front-end* de **SmallL** para o código da máquina virtual **TAM**, de forma a ser possível compilar e executar programas em **SmallL**. O interpretador das instruções de **TAM** está disponível na página desse Curso.

2 Testes

Para testar seu tradutor use como entrada o código a seguir. É importante acrescentar outros testes.

```
1) L1:L3:  i = i + 1
2) L5:     t1 = i * 8
3)         t2 = a [ t1 ]
4)         if t2 < v goto L3
5) L4:     j = j - 1
6) L7:     t3 = j * 8
7)         t4 = a [ t3 ]
8)         if t4 > v goto L4
9) L6:     iffalse i >= j goto L8
10) L9:    goto L2
11) L8:    t5 = i * 8
12)        x = a [ t5 ]
13) L10:   t6 = i * 8
14)        t7 = j * 8
15)        t8 = a [ t7 ]
16)        a [ t6 ] = t8
17) L11:   t9 = j * 8
18)        a [ t9 ] = x
19)        goto L1
20) L2:
```

que representa o código intermediário produzido pelo *front-end* para o seguinte programa fonte:

```
1)  {                // Arquivo test
2)    int i; int j; float v; float x; float[100] a;
3)    while( true ) {
4)       do i = i+1; while( a[i] < v);
5)       do j = j-1; while( a[j] > v);
```

```
6)       if( i >= j ) break;
7)       x = a[i]; a[i] = a[j]; a[j] = x;
8)    }
9)  }
```

3 Sugestão de Implementação do Tradutor em C++

arq: QUADRUPLAS.HPP

```
....

class Quadrupla
{
    private:
        string op;                                  /* operador */
        string ops[NUM_OPERANDOS];                  /* operandos 0, 1 e 2 */

    public:
        Quadrupla();                                /* construtor */
        Quadrupla( string op, string a, string b, string c); /* construtor */
        string get-operador( void );                        /* retorna operador */
        void set-operador( string novo );                   /* muda operador para novo */
        string get-operado-pos( int pos );                    /* retorna operando pos */
        void set-ops( int pos, string v );          /* muda valor do operando pos para v */
        Quadrupla stringToQuadrupla( string s );    /* retorna a quadrupla representada por s *
        void imprimeQuadrupla();
};

....
```

---

arq. QUADRUPLAS.CPP

```
Quadruplas::Quadruplas()
{
    quadruplas.resize(0);
}

Quadrupla Quadruplas::getQuadrupla( int pos )
{
    return quadruplas[pos];
}

void Quadruplas::addQuadrupla( Quadrupla q )
{
    quadruplas.push_back(q);
}

int Quadruplas::getNumQuadruplas()
{
    return quadruplas.size();
}
```

```
void Quadruplas::imprimeQuadruplas()
{
    Quadrupla p;
    cout << "Impressao das quadruplas" << endl;
    cout << "----------------------" << endl;
    for ( unsigned int i = 0; i < quadruplas.size(); i++ )
    {
        q = quadruplas[i];
        q.imprimeQuadrupla();
    }
    cout << "----------------------" << endl << endl;
}
```

---

arq. TABELA DE SIMBOLOS

```
tabela.hpp
tabela.cpp

main.cpp

...

void teste1( Tabela t )
{
    cout << endl << "----- Impressao do Teste 1 ----- " << endl;
    t.entrada_bloco();
    t.instala("a", "integer");
    t.instala("b", "integer");
    t.instala("c", "integer");
    t.imprime();
    t.saida_bloco();
}

void teste2( Tabela t )
{
    cout << endl << "----- Impressao do  Teste 2 ----- " << endl;
    t.entrada_bloco();
    t.instala("i", "INTEGER");
    ...
    t.imprime();
    t.saida_bloco();
}

void teste3( Tabela t )
{
    cout << endl << "----- Impressao do Teste 3 ----- " << endl;
    t.entrada_bloco();
    t.instala("a", "integer");
    ...
    t.imprime();
    t.saida_bloco();
}
```

```
void teste4( Tabela t )
{
    cout << endl << "----- Impressao do Teste 4 ----- " << endl;
    ....
}



void teste5( Tabela t )
{
    cout << endl << "----- Impressao do Teste 5 ----- " << endl;
    ....
}

.....

int main(int argc, char **argv){

    Tabela t;


    teste1(t); t.limpa();
    teste2(t); t.limpa();
    teste3(t); t.limpa();
    teste4(t); t.limpa();
    teste5(t); t.limpa();


return 0;
}


------------------------------------------------------------


arq. TRADUTOR.hpp

....

class Tradutor
{
private:
Tabela tabela;
string instrucao;
vector<string> instrucoesTAM;
map<string, string (*)(Quadrupla &q, Tabela &t)> opcodes;

public:
Tradutor();
string getInstrucao();
void setInstrucao( string s );
void traduzQuadruplas( Quadruplas Q );
string traduzQuadrupla( Quadrupla q );
void adicionaInstrucao( string inst );
        void imprimeProgramaTAM( string file_name );
};
```

....

---

arq. TRADUTOR.cpp

.....

```
void Tradutor::imprimeProgramaTAM( string file_name )
{
    ofstream output( file_name.c_str(), ios::out );

cout << "Impressao do Programa TAM" << endl;
cout << "------------------------" << endl;
for ( unsigned int i = 0; i < instrucoesTAM.size(); i++ )
{
cout << instrucoesTAM[i];
        output << instrucoesTAM[i];
}
cout << "------------------------" << endl;

    output.close();
}
```

---

arq. MAIN.cpp

.....

```
void leArqEntrada( string file_name, Quadruplas &Q )
{
    string s;
    Quadrupla q;

    ifstream input( file_name.c_str(), ios::in );

    if ( !input )
    {
        cout << "O arquivo " << file_name << " nao existe. O programa terminara agora!" << endl
        exit(0);
    }
    else
    {
        while ( getline( input, s ) )
        {
            q = q.stringToQuadrupla( s );
            Q.addQuadrupla( q );
        }
    }

    input.close();
}

int main( int argc, char *argv[] )
```

```
{
    string file_name_in;
    string file_name_out;
    Tradutor T;
    Quadruplas Q;

    if( argc < 3 )
    {
        cout << "Escreva o nome do arquivo de entrada: " << endl;
        cin >> file_name_in;
        cout << "Escreva o nome do arquivo de saida: " << endl;
        cin >> file_name_out;
    }
    else
    {
        file_name_in = argv[1];
        file_name_out = argv[2];
    }

    leArqEntrada( file_name_in, Q );

    cout << endl;
    Q.imprimeQuadruplas();
    T.traduzQuadruplas( Q );
    T.imprimeProgramaTAM( file_name_out );
    cout << endl;

    return 0;
}
```
