

AULA 1 - Correção da Prova

```
namespace MeuCompilador
{
    class GeradorExecutavelBLL
    {
        private static FileStream infile, outfile;

        public static void gravaCodigo(String _arquivo)
        {
            int tam;
            char x;

            infile = new System.IO.FileStream(_arquivo, System.IO.FileMode.Open,
System.IO.FileAccess.Read);
            tam = (int)infile.Length;
            for (int i = 0; i < tam; ++i)
            {
                x = (char)infile.ReadByte();
                outfile.WriteByte((byte)(x));
            }
            infile.Close();
        }
        public static void geraExecutavel()
        {
            outfile = new
System.IO.FileStream("PROGRAMA.COM", System.IO.FileMode.Create, System.IO.FileAccess.Wri
te);

            MeuCompiladorDAL.populaDR();
            MeuCompiladorDAL.leUmTokenValido();
            while (Erro.getErro() == false)
            {
                if (int.Parse(Token.getCodigo()) < 100)
                {
                    gravaCodigo("Codigo" + Token.getCodigo() + ".COM");
                }
                MeuCompiladorDAL.leUmTokenValido();
            }
            Erro.setErro(false);
            outfile.Close();
        }
    }
}
```

AULA 2 - PONTOCOM.LIB

GERADOREXECUTAVEL

```
namespace MeuCompilador
{
    class GeradorExecutavelBLL
    {
        private static FileStream infile, outfile;
```

```

        public static void gravaCodigo()
        {
            int tam;
            char x;

            infile = new System.IO.FileStream("pontocom.lib",
System.IO.FileMode.Open, System.IO.FileAccess.Read);
            MeuCompiladorDAL.consultaIndiceLib();
            infile.Position = int.Parse(IndiceLib.getInicio());
            tam = int.Parse(IndiceLib.getTamanho());
            for (int i = 0; i < tam; ++i)
            {
                x = (char)infile.ReadByte();
                outfile.WriteByte((byte)(x));
            }
            infile.Close();
        }
        public static void geraExecutavel()
        {
            outfile = new
System.IO.FileStream("PROGRAMA.COM",System.IO.FileMode.Create,System.IO.FileAccess.Wri
te);

            MeuCompiladorDAL.populaDR();
            MeuCompiladorDAL.leUmTokenValido();
            while (Erro.getErro() == false)
            {
                if (int.Parse(Token.getCodigo()) <100)
                {
                    gravaCodigo();
                }
                MeuCompiladorDAL.leUmTokenValido();
            }
            Erro.setErro(false);
            outfile.Close();
        }
    }
}
MEUCOMPILADORDAL
public static void consultaIndiceLib()
{
    OleDbDataReader resultInd;
    String aux = "select * from TIndLib where codigo =" + Token.getCodigo();

    strSQL = new OleDbCommand(aux, conn);
    resultInd = strSQL.ExecuteReader();
    Erro.setErro(false);
    if (resultInd.Read())
    {
        IndiceLib.setInicio("'" + resultInd.GetInt32(1));
        IndiceLib.setTamanho("'" + resultInd.GetInt32(2));
    }
}

```

BANCO

Codigo	Inicio	Tamanho	Clique para Adicionar
0	0	16	
1	16	10	
2	26	8	
3	34	2	
*			

- Essa solução se mostra mais eficiente, visto que se aumentar a quantidade de arquivos a serem lidos, só será necessário ler a biblioteca e, com isso evitar ter de escrever o nome de todos os arquivos.

AULA 3 - Tratando os argumentos

```

***** LIMPATELA *****
153C:0100 B406 MOV AH,06
153C:0102 B025 MOV AL,25
153C:0104 B700 MOV BH,07
153C:0106 B500 MOV CH,00
153C:0108 B100 MOV CL,00
153C:010A B618 MOV DH,18
153C:010C B24F MOV DL,4F
153C:010E CD10 INT 10
***** POSICIONACURSOR *****
153C:0110 B60A MOV DH,0A
153C:0112 B228 MOV DL,28
153C:0114 B402 MOV AH,02
153C:0116 B700 MOV BH,00
153C:0118 CD10 INT 10
***** ESCREVEDIGITO *****
153C:011A B053 MOV AL,38
153C:011C B40E MOV AH,0E
153C:011E B700 MOV BH,00
153C:0120 CD10 INT 10
***** FIM *****
153C:0122 CD20 INT 20

```

➔ DH informa a linha, no caso 10
DL informa a coluna, no caso 40

➔ AL informa o caracter a ser exibido,
no caso '8'

```

namespace MeuCompilador
{
    class GeradorExecutavelBLL
    {
        private static FileStream infile, outfile;

        public static void gravaCodigo()
        {
            int tam, qtdargs;
            char x;

            infile = new System.IO.FileStream("pontocom.lib",
            System.IO.FileMode.Open, System.IO.FileAccess.Read);
            MeuCompiladorDAL.consultaIndiceLib();
            infile.Position = int.Parse(IndiceLib.getInicio());
            tam = int.Parse(IndiceLib.getTamanho());
            qtdargs = MeuCompiladorDAL.leQtdArgumentos();

```

```

for (int i = 0; i < tam; ++i)
{
x = (char)infile.ReadByte();
outfile.WriteByte((byte)(x));
if (qtdargs != 0)
{
    MeuCompiladorDAL.leUmTokenValido();
    outfile.WriteByte((byte)int.Parse(Token.getToken()));
    infile.ReadByte();
    ++i;
    --qtdargs;
}
}
infile.Close();
}

```

AULA 4 / 5 - Solucionando o problema com o caracter ASCII

- No TP da Semana02 a saída não correspondia ao programado em teste.txt pois estávamos desprezando os argumentos dos comandos armazenados em TTokensValidos;
- Aí na Semana03 passamos a considerar estes argumentos (solução disponível em Semana04);
- Em nossa solução o PosicionaCursor funciona perfeitamente porém o EscreveDigito ao invés de nos mostrar "18" nos mostra uma carinha sorrindo;
- Isto porque o dígito 1 está sendo considerado como um número enquanto gostaríamos que ele fosse considerado como o char cujo código ASCII é 49 ou 31h;
- Para resolver este problema basta somarmos ao dígito em questão o valor 48 ou 30h antes de gravá-lo como mostramos na sequência.

Solução do Professor

```

aux = int.Parse(Token.getCodigo());
for (int i = 0; i < tam; ++i)
{
x = (char)infile.ReadByte();
outfile.WriteByte((byte)x);
if (qtdargs != 0)
{
    MeuCompiladorDAL.leUmTokenValido();
    if (aux == 2)
        outfile.WriteByte(byte.Parse("'" + (48 + int.Parse(Token.getToken()))));
    else
        outfile.WriteByte(byte.Parse(Token.getToken()));
    x = (char)infile.ReadByte();
    ++i;
    --qtdargs;
}
}

```

- A solução abordada acima “perguntando toda hora” se aux = 2 não é necessária, visto que só é preciso saber se tem um argumento e que o mesmo seja o escrevedigito.
- A solução ideal seria colocar o tipo no banco de dados para cada código, colocando inteiro quando for mais do que um argumento e char quando for somente um, pois assim saberíamos que quando só tivéssemos um argumento, este seria o escrevedigito.

TArgLimites				
Codigo	Posicao	Minimo	Maximo	
1	0	1	80	
1	1	1	24	
2	0	0	9	
*				

TArgLimites					
Codigo	Posicao	Minimo	Maximo	Tipo	
1	0	1	80	i	
1	1	1	24	i	
2	0	0	9	c	
*					

```

public static void gravaCodigo()
{
    int tam, qtdargs, pos;
    String aux;
    char x;

    infile = new System.IO.FileStream("pontocom.lib",
System.IO.FileMode.Open, System.IO.FileAccess.Read);
    MeuCompiladorDAL.consultaIndiceLib();
    infile.Position = int.Parse(IndiceLib.getInicio());
    tam = int.Parse(IndiceLib.getTamanho());
    qtdargs = MeuCompiladorDAL.leQtdArgumentos();
    pos = 0;
    aux = Token.getCodigo();
    for (int i = 0; i < tam; ++i)
    {
        x = (char)infile.ReadByte();
        outfile.WriteByte((byte)(x));
        if (qtdargs != 0)
        {
            MeuCompiladorDAL.leUmTokenValido();
            ArgLim.setCodigo(aux);
            ArgLim.setposicao("" + pos);
            MeuCompiladorDAL.leUmLimite();
            if (ArgLim.gettipo().Equals("c"))

```

```

        outfile.WriteByte(byte.Parse(""));
    }
    else
    {
        outfile.WriteByte(byte.Parse(Token.getToken()));
        infile.ReadByte();
        ++i;
        ++pos;
        --qtdargs;
    }
}
infile.Close();
}

```

MEUCOMPILADOR COMPLETO

Classe Token tem:

- código, token, tipo e linha;

Classe IndiceLib tem:

- código, início e tamanho;

Classe Gabarito tem:

- code,prior,info e next;

Classe ArgLim tem:

- codigo, posicao, minimo, maximo e tipo;

MEUCOMPILADOR DAL

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;

namespace MeuCompilador
{
    class MeuCompiladorDAL
    {
        private static String strConexao = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Compilador.mdb";
        private static OleDbConnection conn = new OleDbConnection(strConexao);
        private static OleDbCommand strSQL;
        private static OleDbDataReader resultAS, resultgab, resultqtdarg,
resultlimites;

        public static void conecta()
        {
            Erro.setErro(false);

```

```

        try
        {
            conn.Open();
        }
        catch
        {
            Erro.setErro("A conexão falhou!");
            return;
        }
    }

    public static void desconecta()
    {
        conn.Close();
    }

    public static void inseriUmTokenValido()
    {
        String aux = "insert into TTokensValidos(codigo,token,tipo,linha) values
(" + Token.getCodigo() + ",'" + Token.getToken() + "','" + Token.getTipo() + "','" +
Token.getLinha() + ")";

        strSQL = new OleDbCommand(aux, conn);
        strSQL.ExecuteNonQuery();
    }

    public static void deletaTTokensValidos()
    {
        String aux = "delete * from TTokensValidos";

        strSQL = new OleDbCommand(aux, conn);
        strSQL.ExecuteNonQuery();
    }

    public static void deletaDelimitadores()
    {
        String aux = "delete * from TTokensValidos where Tipo = 'Delimitador'";

        strSQL = new OleDbCommand(aux, conn);
        strSQL.ExecuteNonQuery();
    }

    public static void consultaUmToken()
    {
        OleDbDataReader result;
        String aux = "select * from TTokens where Token =' " + Token.getToken() +
""";

        strSQL = new OleDbCommand(aux, conn);
        result = strSQL.ExecuteReader();
        Erro.setErro(false);
        if (result.Read())
            Token.setCodigo("" + result.GetInt32(0));
        else

```

```

        Erro.setErro("Linha " + Token.getLinha() + ": " + Token.getToken() + "
(Token não identificado)");
    }

    public static void consultaIndiceLib()
    {
        OleDbDataReader resultInd;
        String aux = "select * from TIndLib where codigo =" + Token.getCodigo();

        strSQL = new OleDbCommand(aux, conn);
        resultInd = strSQL.ExecuteReader();
        Erro.setErro(false);
        if (resultInd.Read())
        {
            IndiceLib.setInicio("'" + resultInd.GetInt32(1));
            IndiceLib.setTamanho("'" + resultInd.GetInt32(2));
        }
    }

    public static void populaDR()
    {
        String aux = "select * from TTokensValidos";

        strSQL = new OleDbCommand(aux, conn);
        resultAS = strSQL.ExecuteReader();
    }

    public static void leUmTokenValido()
    {
        Erro.setErro(false);
        if (resultAS.Read())
        {
            Token.setCodigo("'" + resultAS.GetInt32(0));
            Token.setToken(resultAS.GetString(1));
            Token.setTipo(resultAS.GetString(2));
            Token.setLinha("'" + resultAS.GetInt32(3));
        }
        else
            Erro.setErro(true);
    }

    public static void populaGabarito(int _code)
    {
        String aux = "select * from gabarito where code = " + _code;

        strSQL = new OleDbCommand(aux, conn);
        resultgab = strSQL.ExecuteReader();
    }

    public static void leGabarito()
    {
        Erro.setErro(false);
        if (resultgab.Read())
        {

```



```

        Gabarito.setCode("" + resultgab.GetInt32(0));
        Gabarito.setPrior(resultgab.GetString(1));
        Gabarito.setInfo("" + resultgab.GetInt32(2));
        Gabarito.setNext(resultgab.GetString(3));
    }
    else
        Erro.setErro(true);
}

public static int leQtdArgumentos()
{
    String aux = "select * from TQtdArgumentos where Codigo =" +
Token.getCodigo();

    strSQL = new OleDbCommand(aux, conn);
    resultqtdarg = strSQL.ExecuteReader();
    resultqtdarg.Read();
    return resultqtdarg.GetInt32(1);
}

public static void leUmLimite()
{
    String aux = "select * from TArgLimites where codigo =" +
ArgLim.getCodigo() + " and posicao =" + ArgLim.getposicao();

    strSQL = new OleDbCommand(aux, conn);
    resultlimites = strSQL.ExecuteReader();
    if (resultlimites.Read())
    {
        ArgLim.setminimo("" + resultlimites.GetInt32(2));
        ArgLim.setmaximo("" + resultlimites.GetInt32(3));
        ArgLim.settipo(resultlimites.GetString(4));
    }
}
}
}
}

```

MEUCOMPILADORBLL

```

class MeuCompiladorBLL
{
    public static void compilarPrograma()
    {
        // Procedimento para Análise Léxica do meu código

        AnalisadorLexicoBLL.filtro();
        AnalisadorLexicoBLL.scanner();
        if (Erro.getErro()) return;

        // Procedimento para Análise Sintática do meu código

        AnalisadorSintaticoBLL.analiseSintatica();
        if (Erro.getErro()) return;
    }
}

```

```

        // Procedimento para Análise Semântica do meu código

        AnalisadorSemanticoBLL.analiseSemantica();
        if (Erro.getErro()) return;

        // Procedimento para gerar o código executável
        GeradorExecutavelBLL.geraExecutavel();
    }
}

```

ANALISADOR SINTÁTICO

```

class AnalisadorSintaticoBLL
{
    public static void validaSequencia()
    {
        Erro.setErro(false);
        MeuCompiladorDAL.populaGabarito(int.Parse(Token.getCodigo()));
        MeuCompiladorDAL.leGabarito();
        if (Erro.getErro())
        {
            Erro.setErro("Linha " + Token.getLinha() + "- token inesperado: " +
Token.getToken());
            return;
        }

        while (!Erro.getErro())
        {
            if (Token.getCodigo() != Gabarito.getInfo())
            {
                Erro.setErro("Linha " + Token.getLinha() + "- token inesperado: "
+ Token.getToken());
                return;
            }
            if (Gabarito.getNext() != "eof") MeuCompiladorDAL.leUmTokenValido();
            MeuCompiladorDAL.leGabarito();
        }
        Erro.setErro(false);
    }

    public static void analiseSintatica()
    {
        MeuCompiladorDAL.populaDR();

        MeuCompiladorDAL.leUmTokenValido();
        while (Erro.getErro() == false)
        {
            validaSequencia();
            if (Erro.getErro()) return;
            MeuCompiladorDAL.leUmTokenValido();
        }
        Erro.setErro(false);
    }
}

```

```

    }
}

```

ANALISADORSEMANTICOBL

```

namespace MeuCompilador
{
    class AnalisadorSemanticoBL
    {
        public static void analiseSemantica()
        {
            Erro.setErro(false);
            MeuCompiladorDAL.deletaDelimitadores();
            MeuCompiladorDAL.populaDR();
            MeuCompiladorDAL.leUmTokenValido();
            while (Erro.getErro() == false)
            {
                int aux = MeuCompiladorDAL.leQtdArgumentos();
                for (int i = 0; i < aux; ++i)
                {
                    ArgLim.setCodigo(Token.getCodigo());
                    ArgLim.setposicao("" + i);
                    MeuCompiladorDAL.leUmlimite();
                    MeuCompiladorDAL.leUmTokenValido();
                    if (int.Parse(Token.getToken()) < int.Parse(ArgLim.getminimo())
|| int.Parse(Token.getToken()) > int.Parse(ArgLim.getmaximo()))
                    {
                        Erro.setErro("Linha " + Token.getLinha() + ": valor fora da faixa
(" + Token.getToken() + ")");
                        return;
                    }
                }
                MeuCompiladorDAL.leUmTokenValido();
            }
            Erro.setErro(false);
        }
    }
}

```

ANALISADORLÉXICO

```

class AnalisadorLexicoBL
{
    public static void filtro()
    {
        FileStream infile, outfile;
        int tam;
        char x;

        infile = new System.IO.FileStream(ProgramaFonte.getPathNome(),
                                            System.IO.FileMode.Open,
                                            System.IO.FileAccess.Read);
        outfile = new System.IO.FileStream("pftmp.txt",
                                            System.IO.FileMode.Create,
                                            System.IO.FileAccess.Write);
    }
}

```

```

        tam = (int)infile.Length;
    for (int i = 0; i < tam; ++i)
    {
        x = (char)infile.ReadByte();
        if (x == '#')
        {
            ++i;
            do
            {
                x = (char)infile.ReadByte();
                ++i;
            }
            while (x != '#');
        }
        else
            if (x != ' ')
                outfile.WriteByte((byte)char.ToUpper(x));
    }
    infile.Close();
    outfile.Close();
}

public static void scanner()
{
    FileStream infile;
    int tam;
    int kl = 1;
    char x;
    String aux="";

    infile = new System.IO.FileStream("pftmp.txt",
                                      System.IO.FileMode.Open,
                                      System.IO.FileAccess.Read);

    MeuCompiladorDAL.deletaTokensValidos();
    tam = (int)infile.Length;
    for (int i = 0; i < tam; ++i)
    {
        x = (char)infile.ReadByte();

        if (char.IsDigit(x))
        {
            while (char.IsDigit(x))
            {
                aux = aux + x;
                x = (char)infile.ReadByte();
                ++i;
            }
            Token.setCodigo("200");
            Token.setToken(aux);
            Token.setTipo("Inteiro");
            Token.setLinha(kl.ToString());
            MeuCompiladorDAL.inseriUmTokenValido();

```

```

        aux = "";
    }

    if (char.IsLetter(x))
    {
        aux = "";
        while (char.IsLetter(x))
        {
            aux = aux + x;
            x = (char)infile.ReadByte();
            ++i;
        }
        Token.setToken(aux);
        Token.setTipo("String");
        Token.setLinha(kl.ToString());
        MeuCompiladorDAL.consultaUmToken();
        if (Erro.getErro())
            return;
        else
            MeuCompiladorDAL.inseriUmTokenValido();
        aux = "";
    }

    if (char.IsPunctuation(x) || char.IsSymbol(x))
    {
        Token.setToken(x.ToString());
        Token.setTipo("Delimitador");
        Token.setLinha(kl.ToString());
        MeuCompiladorDAL.consultaUmToken();
        if (Erro.getErro())
            return;
        else
            MeuCompiladorDAL.inseriUmTokenValido();
    }

    if (x == 13) ++kl;
}
infile.Close();
}

}

```