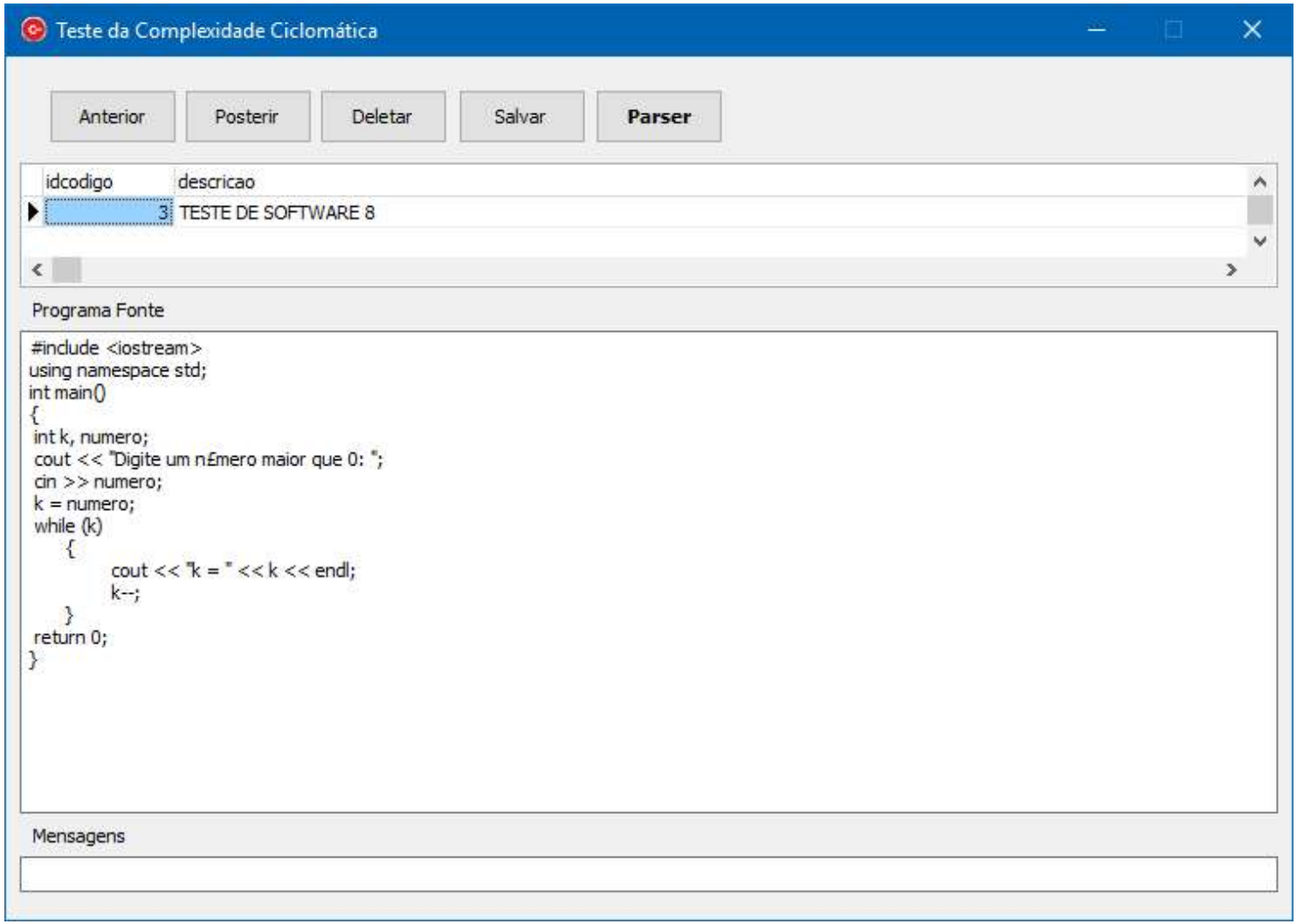
Implementação da Ferramenta de Teste de Software

Complexidade ciclomática, Caminho independente, Grafo de fluxo

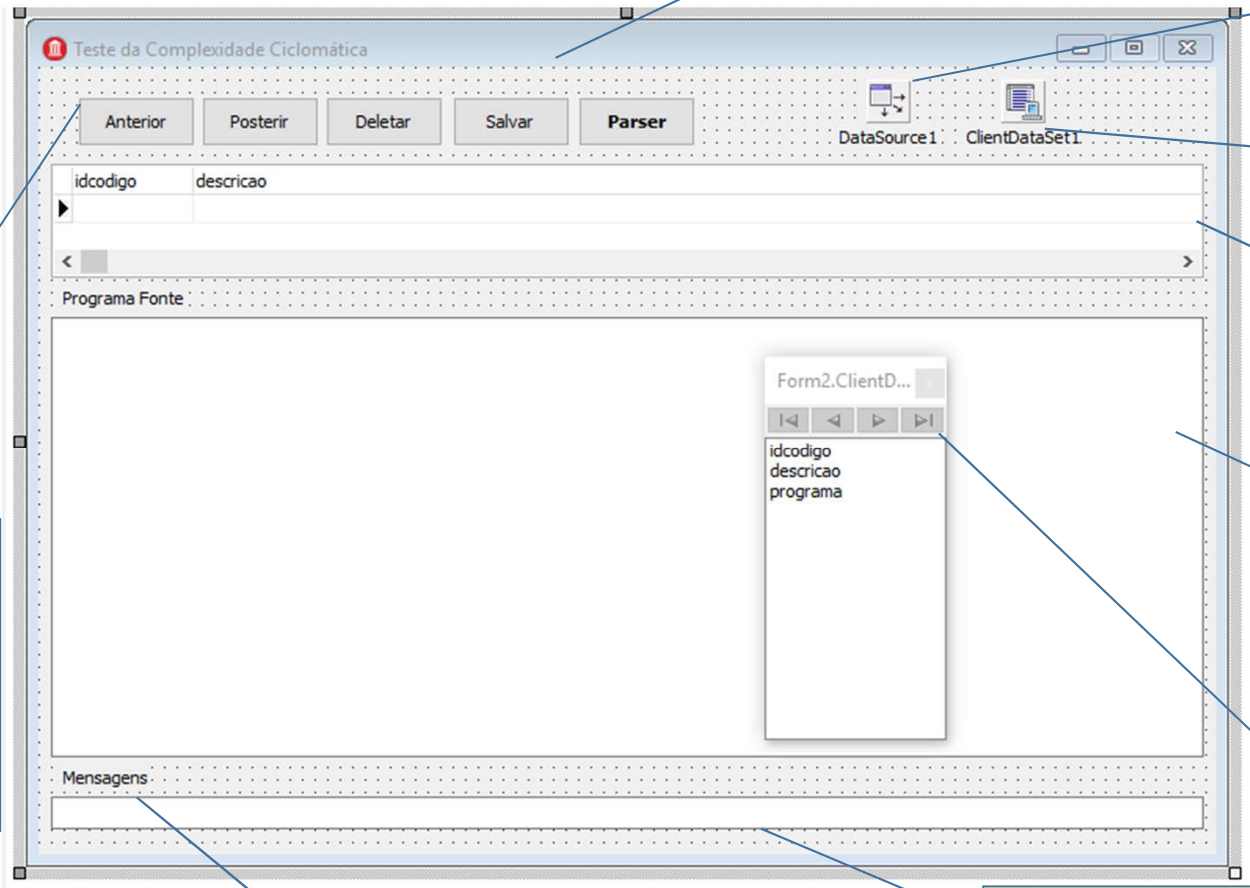
Características da Aplicação

- Teste estrutural (caixa branca) baseado na complexidade ciclomática, referência McCabe (1976);
- Recurso de teste do caminho independente, teste grafo de fluxo;
- Aplicação de processamento local, desktop, para Windows, em interface programada em C++ com Visual Class Library (VCL), Ide Rad Studio;
- Armazenamento de dados por documentos formatados em XML;
- Geração de projeto de casos de testes com código, descrição do projeto e código de fonte;
- Geração de módulo executável.

Interface da aplicação (protótipo da versão 1.0)



Projeto baseado em componentes
Graphical User Interface



Form2
(Form Application)

DataSource1

ClientDataSet1

DBGrid1

Memo1

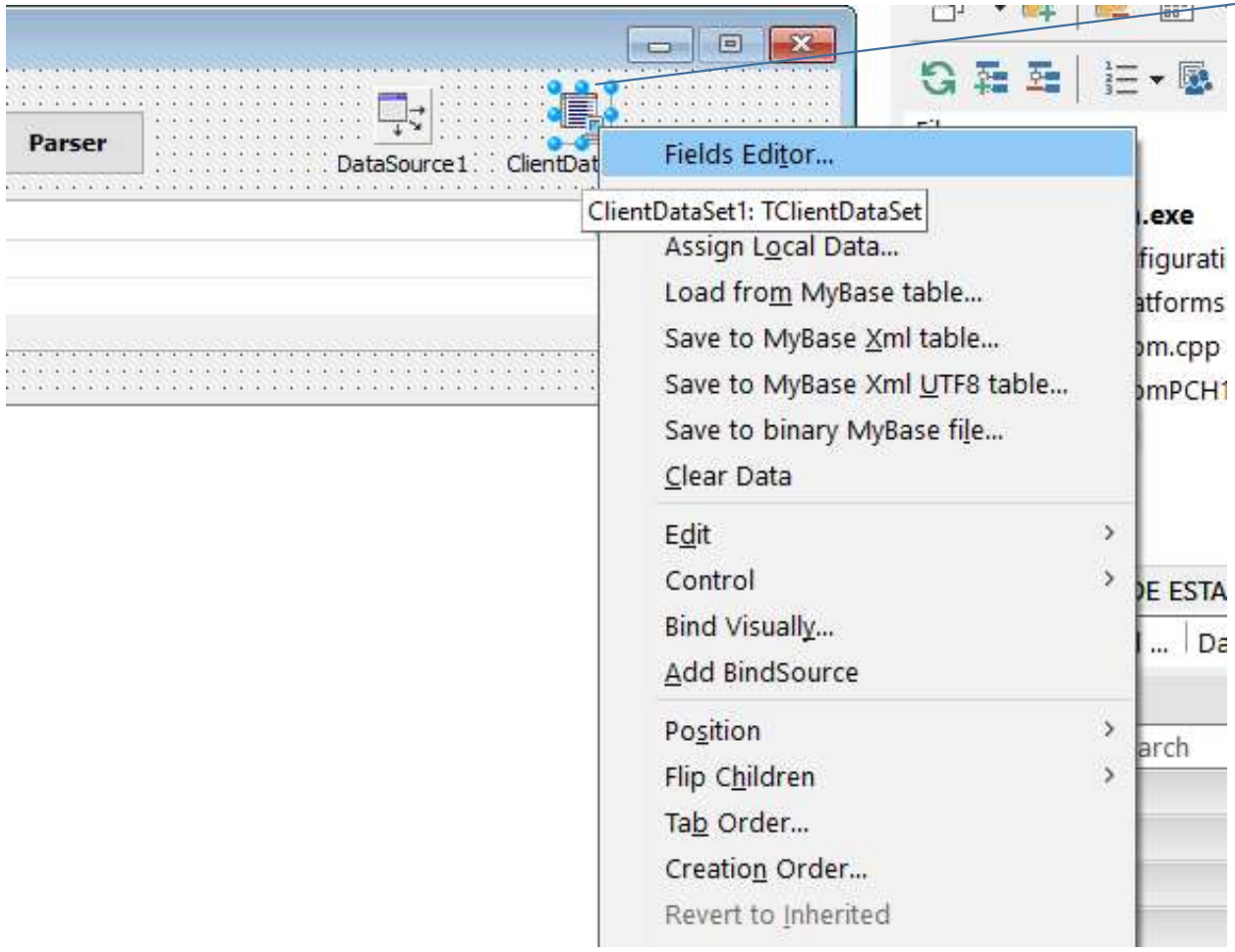
Fields Editor
do
ClientDataSet1

Edit1

Label1 (Programa Fonte),
Label2 (Mensagens)

Button1,
Button2,
Button3,
Button4,
Button5

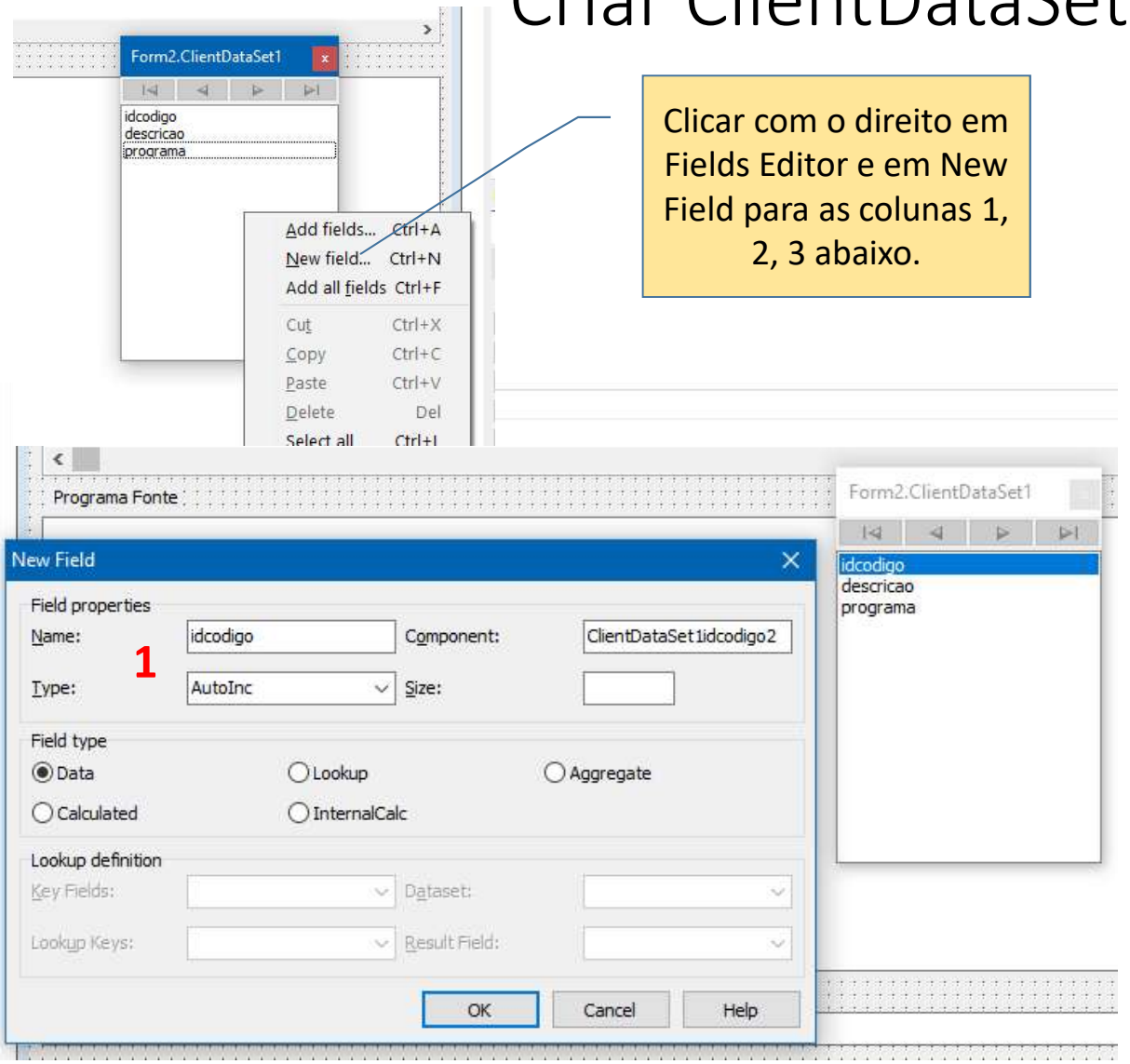
Criar ClientDataSet1 (1/3)

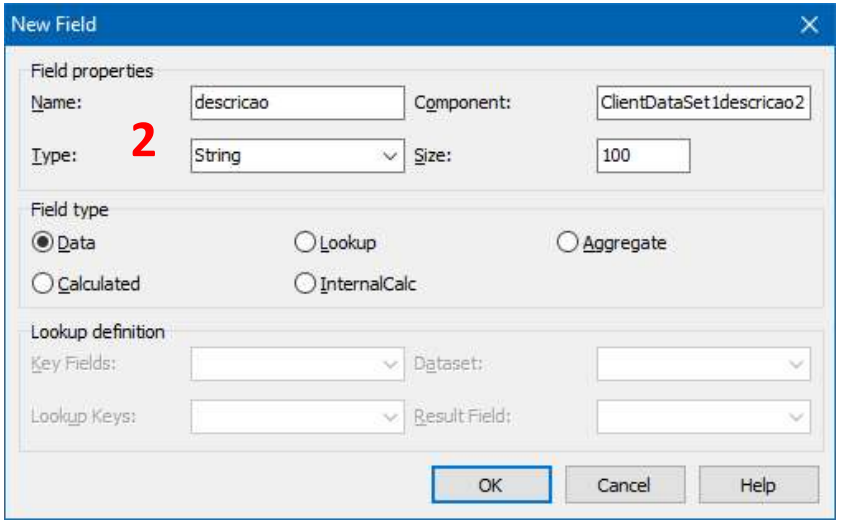


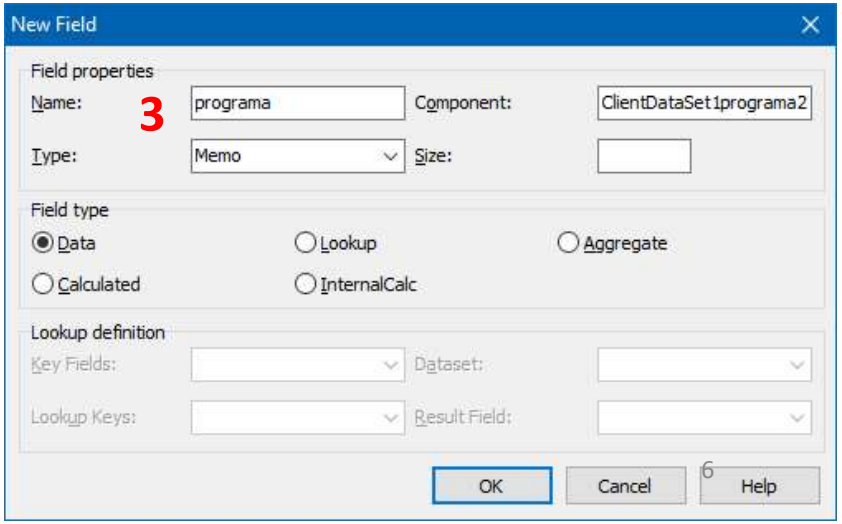
(Criar colunas do ClientDataSet1)
Clicar com o direito do mouse e em Fields Editor

Criar ClientDataSet1 (2/3)

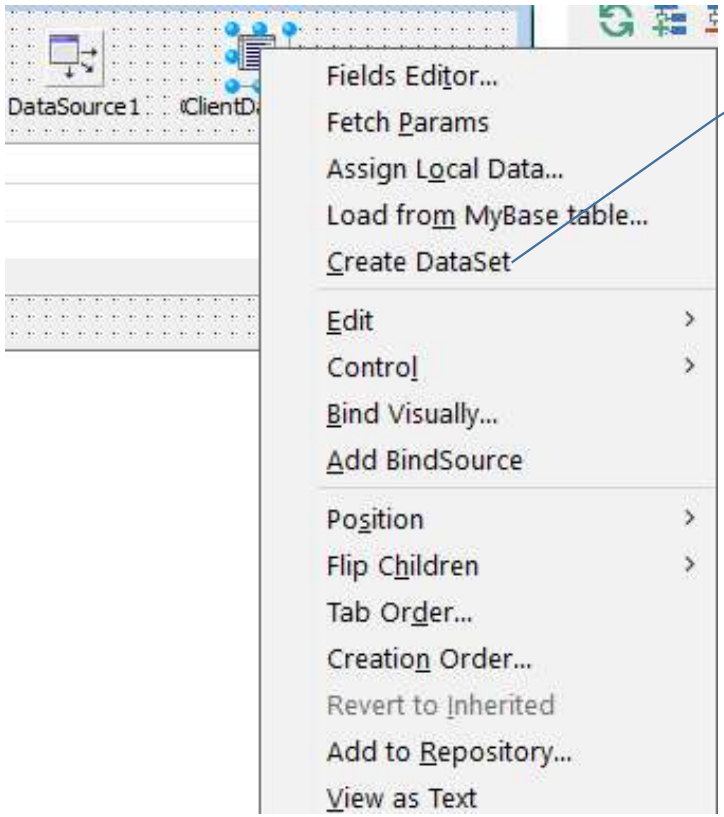
Clicar com o direito em Fields Editor e em New Field para as colunas 1, 2, 3 abaixo.







Criar ClientDataSet1 (3/3)

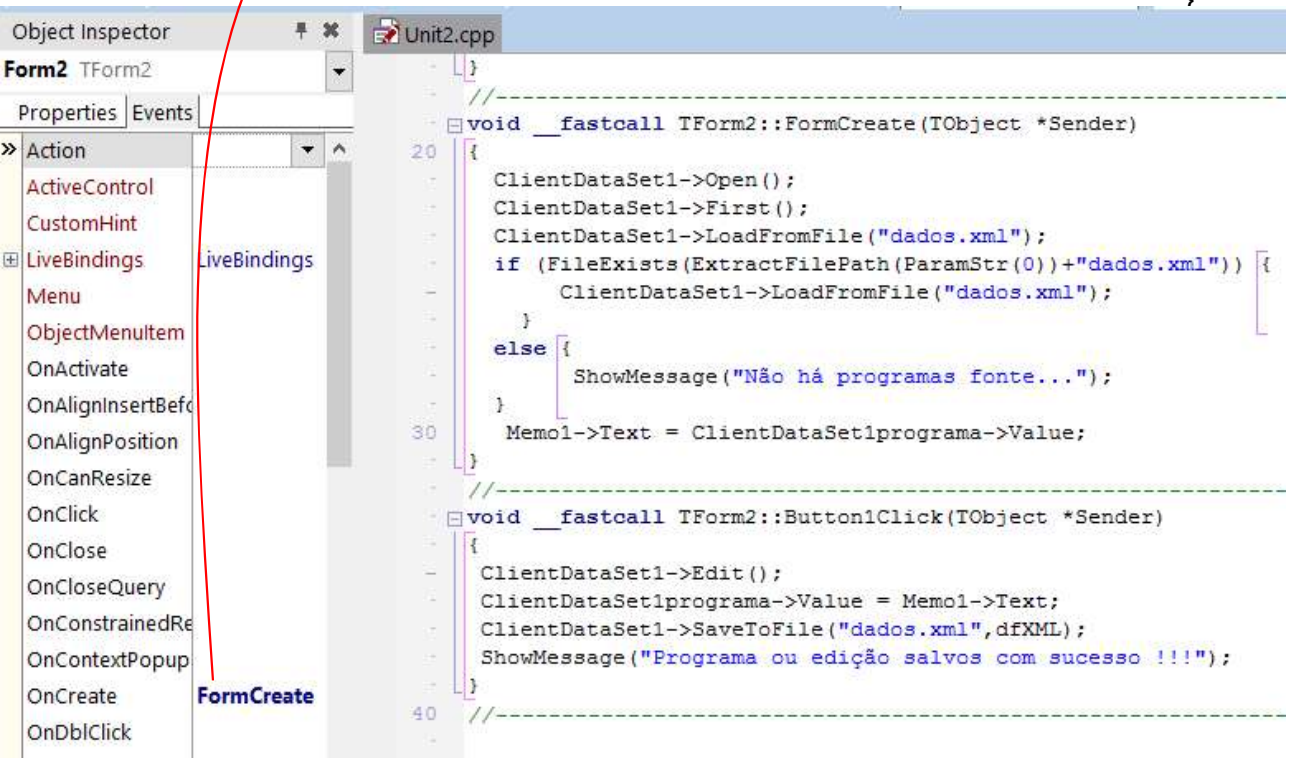


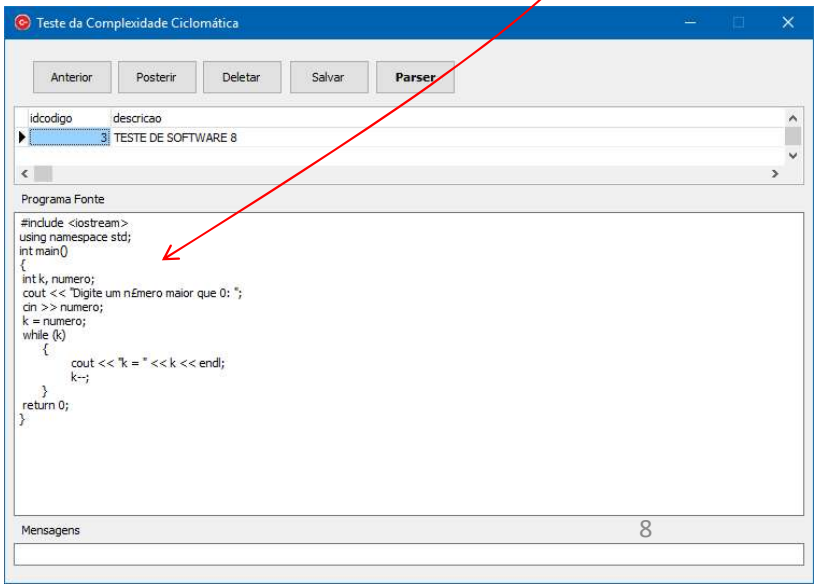
Após criar as colunas do ClientDataSet1, clique com o direito no mesmo componente e depois em Create DataSet

Evento FormCreate

Inicializa a interface com o último código fonte

```
void __fastcall TForm2::FormCreate(TObject *Sender)
{
 ClientDataSet1->Open();
 ClientDataSet1->First();
 ClientDataSet1->LoadFromFile("dados.xml");
 if (FileExists(ExtractFilePath(ParamStr(0))+"dados.xml")) {
                ClientDataSet1->LoadFromFile("dados.xml"); }
 else {
    ShowMessage("Não há programas fonte...");
 }
  Memo1->Text = ClientDataSet1programa->Value;
}
```



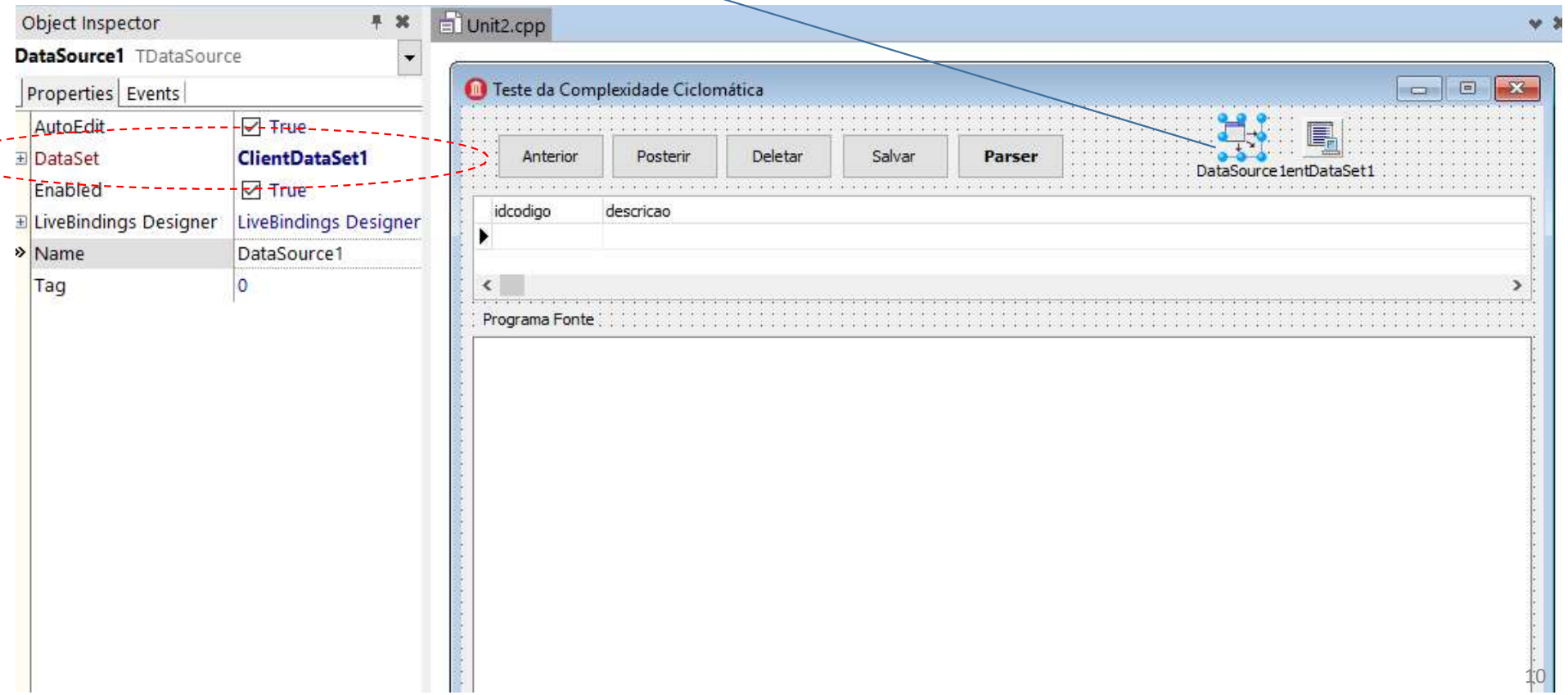


Método carregar registro

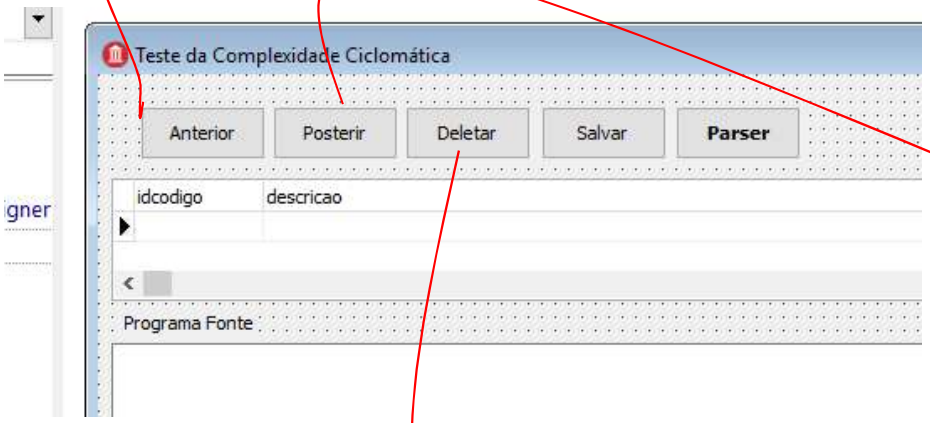
```
void __fastcall TForm2::FormCreate(TObject *Sender)
{
  ClientDataSet1->Open();   //abrir o ClientDataSet1
  ClientDataSet1->First();   //ir para o primeiro registro

  if (FileExists(ExtractFilePath(ParamStr(0))+"dados.xml")) {  //caso não exista dados.xml na pasta do executável, criar aquivo
                ClientDataSet1->LoadFromFile("dados.xml"); }
  else {
      ShowMessage("Não há programas fonte...");
  }
   Memo1->Text = ClientDataSet1programa->Value;  // carregar programa fonte no Memo1
}
```

DataSource1

Atribuir o objeto ClienteDataSet1 à propriedade DataSet de DataSource1
DataSource1->DataSet = ClientDataSet1

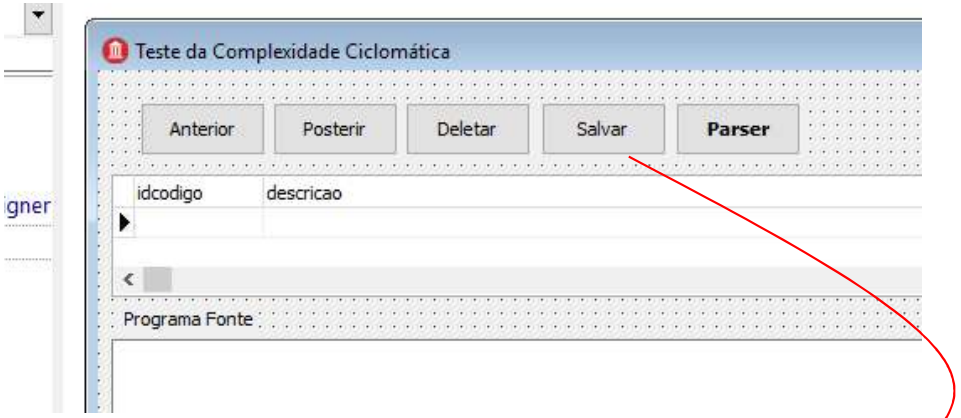


Eventos de Button1, Button2, Button3



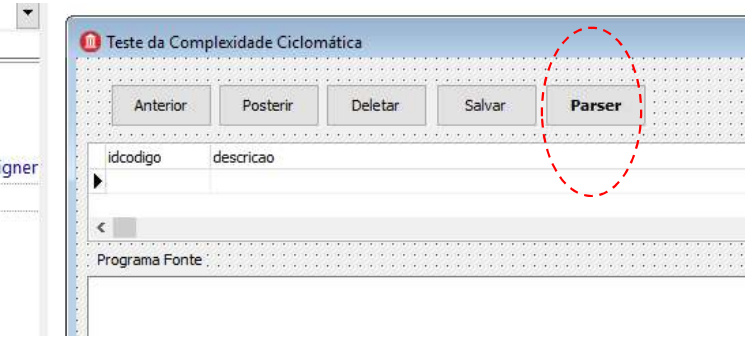

```
void __fastcall TForm2::Button3Click(TObject *Sender)
{
  ClientDataSet1->Prior();
  Memo1->Text = ClientDataSet1programa->Value;
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button4Click(TObject *Sender)
{
  ClientDataSet1->Next();
  Memo1->Text = ClientDataSet1programa->Value;
}
//---------------------------------------------------------------------------


void __fastcall TForm2::Button6Click(TObject *Sender)
{
  ClientDataSet1->Delete();
  ShowMessage("Programa deletado...");

}
```

Evento de Button4 (Salvar)




```
void __fastcall TForm2::Button1Click(TObject *Sender)
{
 ClientDataSet1->Edit(); // coloca o ClientDataSet1 em modo Edição
 ClientDataSet1programa->Value = Memo1->Text; // Grava o código em Memo1 no arquivo dados.xml, na coluna programa
 ClientDataSet1->SaveToFile("dados.xml",dfXML); // salva os registro do ClientDataSet1 em dados.xml
 ShowMessage("Programa ou edição salvos com sucesso !!!");
}
```

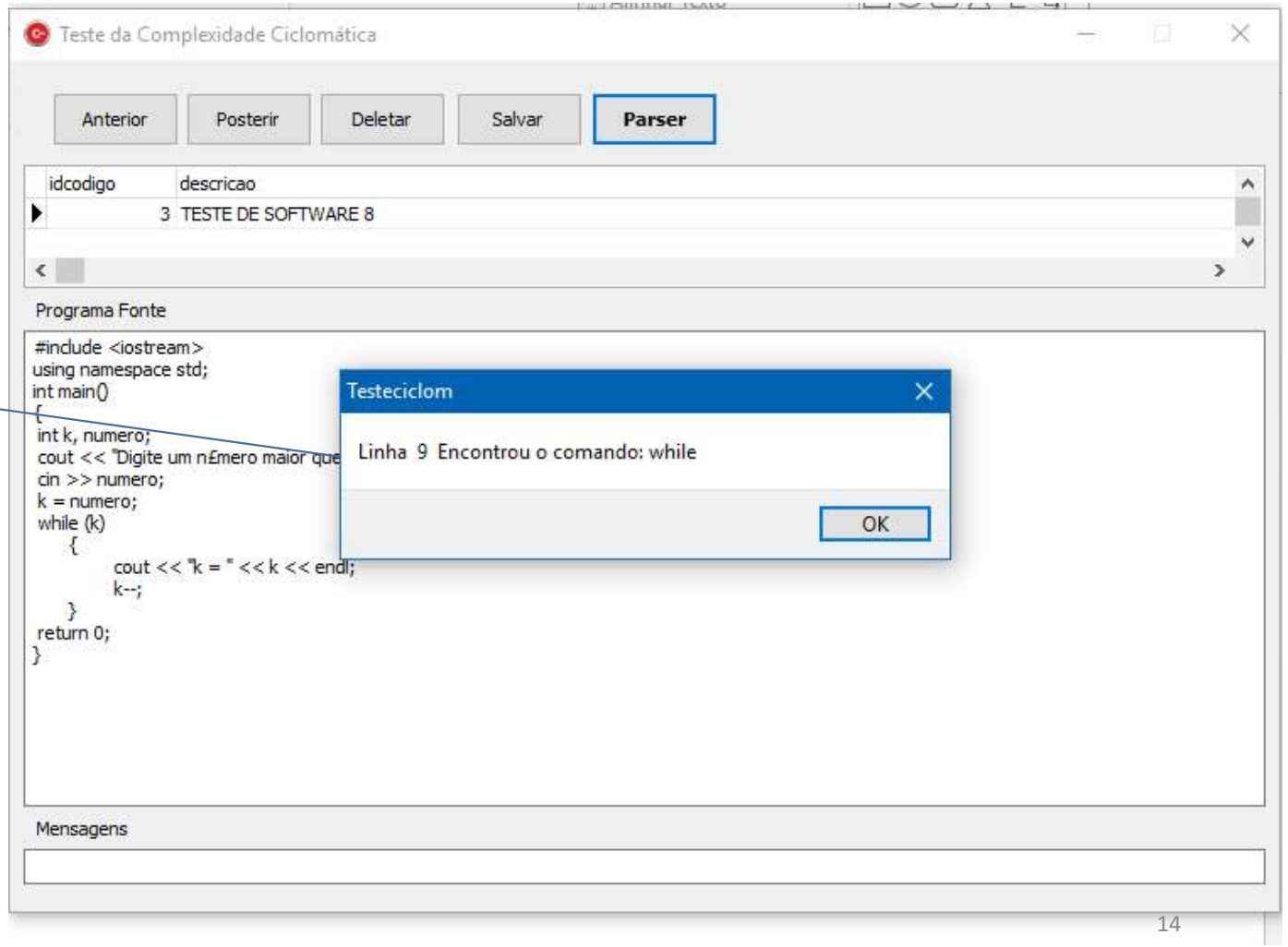
Evento de Button5 (Parser)




```
void __fastcall TForm2::Button2Click(TObject *Sender){
 String mensagem;
 Memo1->Lines->Strings[0]; // vai para primeira linha do Memo1
 AnsiString str;
 for (int i = 0; i < Memo1->Lines->Count-1; i++) { // até a última linha do Memo1
                str = Trim(Memo1->Lines->Strings[i]); // ler cada linha
                str = processo(str); //verifica comando do código (while, for, if, do-while, switch)
                ShowMessage(mensagem= "Linha  " + IntToStr(i+1) +"  Encontrou o comando: " +
str);
                Edit1->Text = str;
  }
  ShowMessage(mensagem);
}

AnsiString processo(AnsiString str){
  AnsiString str_while, str_for, str_if, str_retorno;

 str_while = str.SubString(str.Pos("while"), 5); // caso seja while indica o comando e a linha
 if ( CompareStr("while", str_while) == 0 )
              str_retorno = str_while;
str_for = str.SubString(str.Pos("for"), 3); // caso seja for indica o comando e a linha
 if ( CompareStr("for", str_for) == 0 )
               str_retorno = str_for;
str_if = str.SubString(str.Pos("if"), 2); // caso seja if indica o comando e a linha
 if ( CompareStr("if", str_if) == 0 )
               str_retorno = str_if;
 return str_retorno;  }
```

Parser da Aplicação

Parser no código fonte. Caso encontre comando de ação fundamental (com sentença lógica), indica o número da linha e o valor do comando

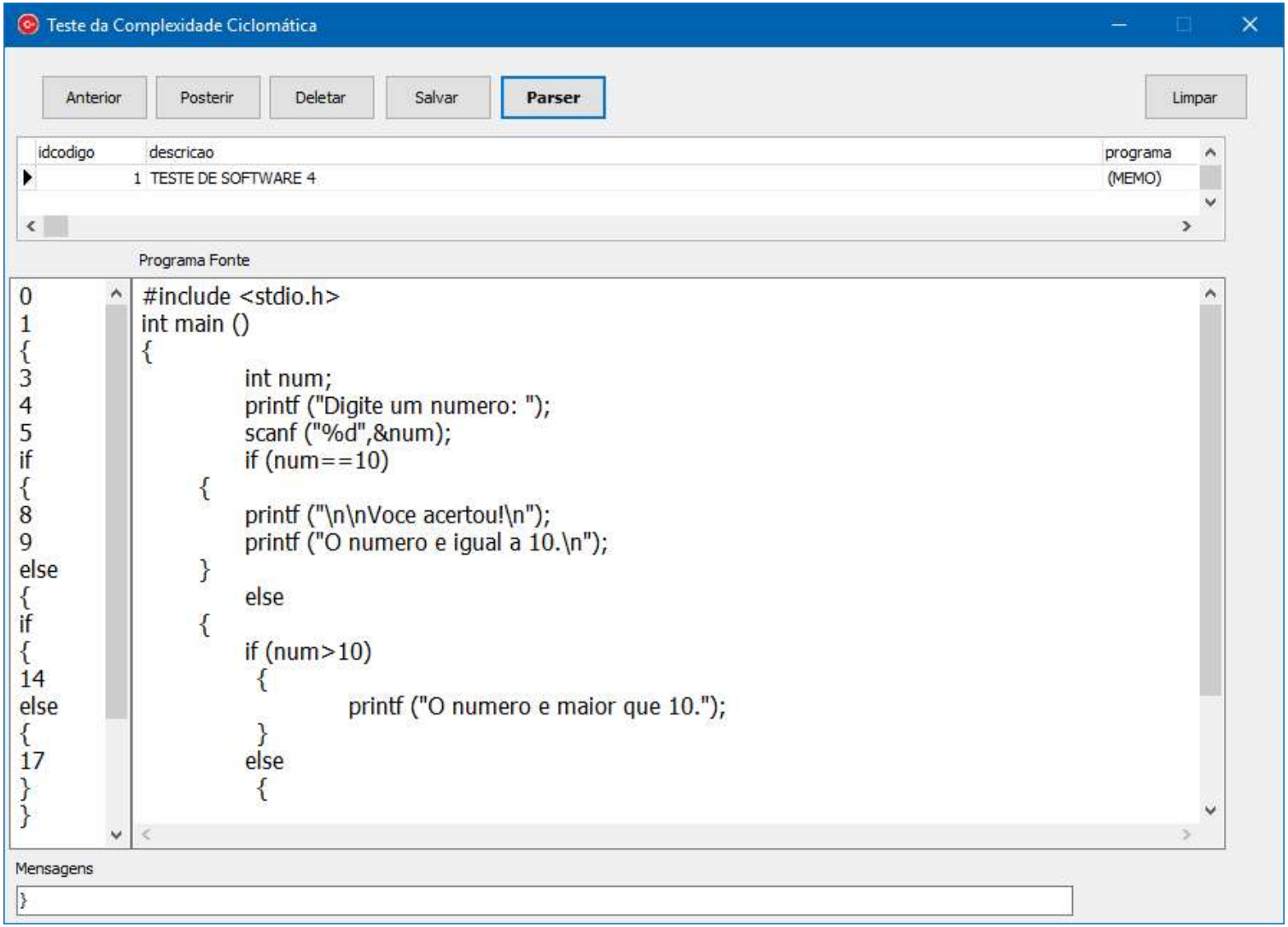


Esta é uma etapa inicial do desenvolvimento.
Para contagem das ações complexas do código, deve-se utilizar os valores resultantes do parser para contagem das complexidades cliclomáticas, determinação do caminho independente pelas fórmulas de McCabe (1976) e compor a matriz de adjacências para o grafo do fluxo.

Implementar resultados pelas tabelas de ranking de risco e probabilidade de falhas pela execução do código fonte. Usar tabelas e gráficos plotados.

PARTE II – Implementação Teste do Caminho Independente e Teste Fluxo do Grafo

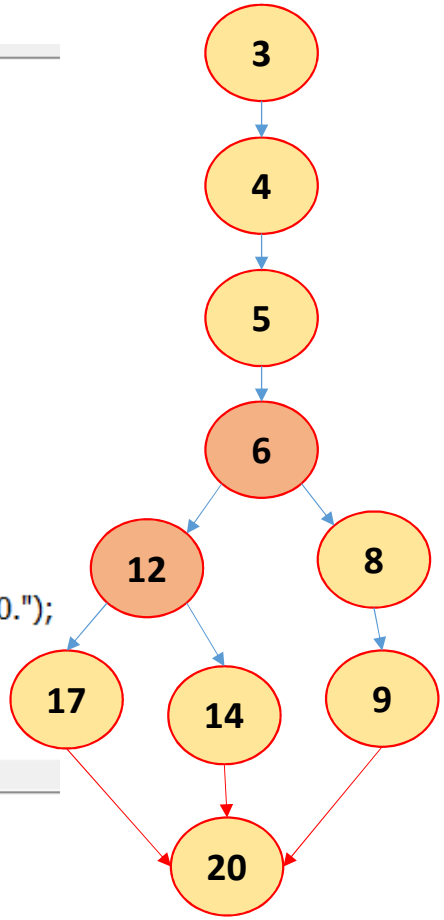
Parser no código para determinar fluxos de execução



Parser no código

```
0  #include <stdio.h>
1  int main ()
{  {
3          int num;
4          printf ("Digite um numero: ");
5          scanf ("%d",&num);
if         if (num==10)
{      {
8          printf ("\n\nVoce acertou!\n");
9          printf ("O numero e igual a 10.\n");
else   }
{          else
if     {
{          if (num>10)
14          {
else                printf ("O numero e maior que 10.");
{           }
17         else
}           {
}
```

Grafo



Matriz de tokens

Linha	Coluna	Token	Obs
3	4	1	seq
4	5	1	seq
5	6	1	seq
6	{	2	If (v)
6	8	1	seq
8	9	1	seq
9	else	5	volta if
6	12	1	else
12	{	2	If (v)
12	14	1	seq
14	else	5	volta if
12	17	1	seq
17	}	3	end if
17	}	4	fim
17	20	4	fim
18	}	4	fim
14	20	1	seq
9	20	1	seq

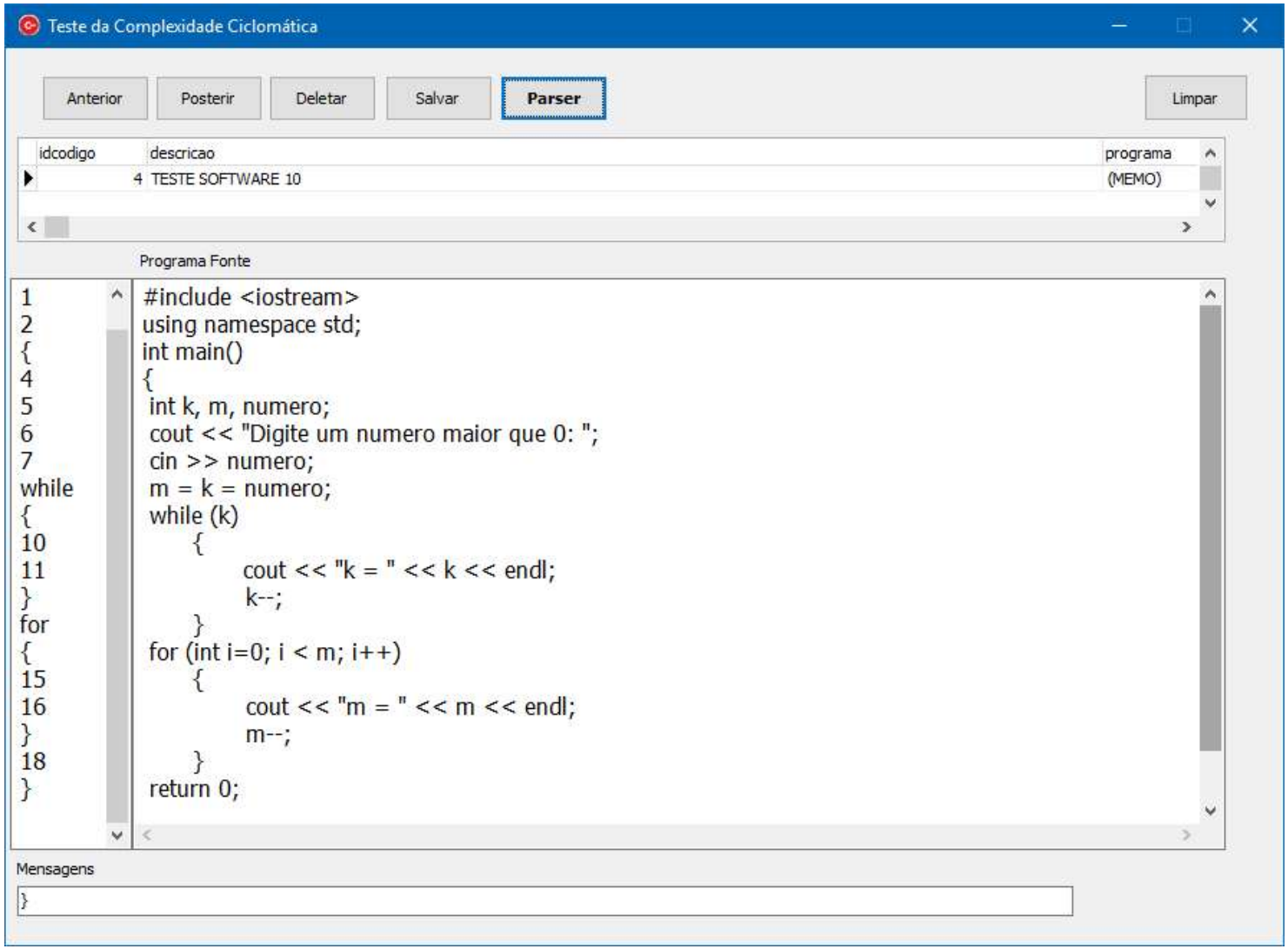
Linha	Coluna	Token	Obs
3	**4**	1	seq
4	**5**	1	seq
5	**6**	1	seq
6	{	2	If (v)
6	**8**	1	seq
8	**9**	1	seq
9	else	5	volta if
6	**12**	1	else
12	{	2	If (v)
12	**14**	1	seq
14	else	5	volta if
12	**17**	1	seq
17	}	3	end if
17	}	4	fim
17	**20**	4	fim
18	}	4	fim
14	**20**	1	seq
9	**20**	1	seq

Matriz de Adjacências

$$m_{i,j} = \begin{cases} 1, se\ i\ é\ adjacente\ a\ j \\ 0, em\ caso\ contrário \end{cases}$$

0	**3**	**4**	**5**	**6**	**8**	**9**	**12**	**14**	**17**	**20**
3		1								
4			1							
5				1						
6					1		1			
8						1				
9										1
12								1	1	
14										1
17										1
20										

Parser no código com estrutura de repetição

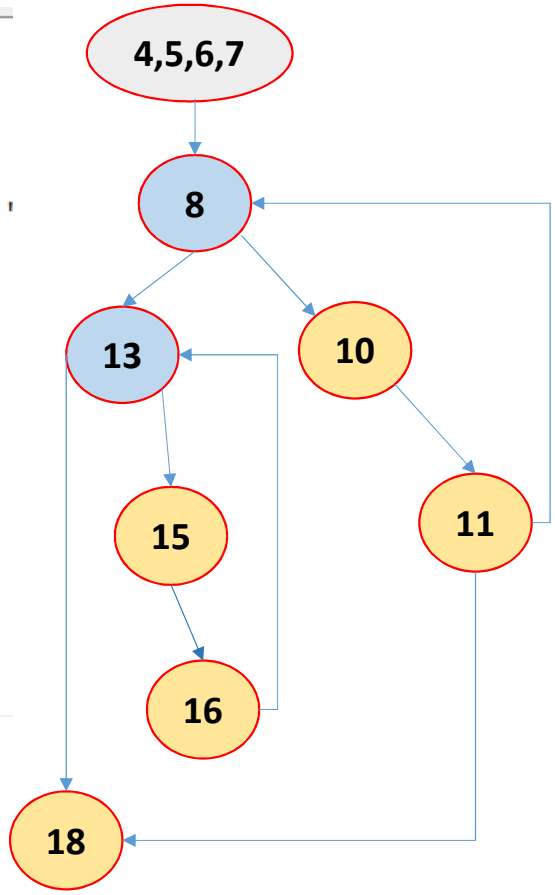


Parser no código

repetições

```
1       #include <iostream>
2       using namespace std;
{       int main()
4       {
5        int k, m, numero;
6        cout << "Digite um numero maior que 0: "
7        cin >> numero;
while    m = k = numero;
{        while (k)
10           {
11                cout << "k = " << k << endl;
}                 k--;
for          }
{        for (int i=0; i < m; i++)
15           {
16                cout << "m = " << m << endl;
}                 m--;
18           }
}        return 0;
```

Grafo



Matriz de tokens

Linha	Coluna	Token	Obs
4	5	1	seq
5	6	1	seq
6	7	1	seq
7	8	1	seq
8	{	6	while (v)
8	10	1	seq
10	11	1	seq
11	}	5	volta while
11	8	1	seq
8	13	1	seq
13	{	7	for (v)
13	15	1	seq
15	16	1	seq
16	}	5	volta for
16	13	1	seq
13	18	4	fim
11	18	4	fim

Matriz de tokens

Linha	Coluna	Token	Obs
4	**5**	1	seq
5	**6**	1	seq
6	**7**	1	seq
7	**8**	1	seq
8	{	6	while (v)
8	**10**	1	seq
10	**11**	1	seq
11	}	5	volta while
11	**8**	1	seq
8	**13**	1	seq
13	{	7	for (v)
13	**15**	1	seq
15	**16**	1	seq
16	}	5	volta for
16	**13**	1	seq
13	**18**	4	fim
11	**18**	4	fim

Matriz de Adjacências

0	**4**	**5**	**6**	**7**	**8**	**10**	**11**	**13**	**15**	**16**	**18**
4		1									
5			1								
6				1							
7					1						
8						1		1			
10							1				
11					1						1
13									1		1
15										1	
16								1			
18											

1) Algoritmo para gerar matriz de adjacentes, segundo o algoritmo no
Livro de Nivil Ziviane, Projeto de Algoritmos em Java e C++

2) Ver código do parser no projeto do aplicativo