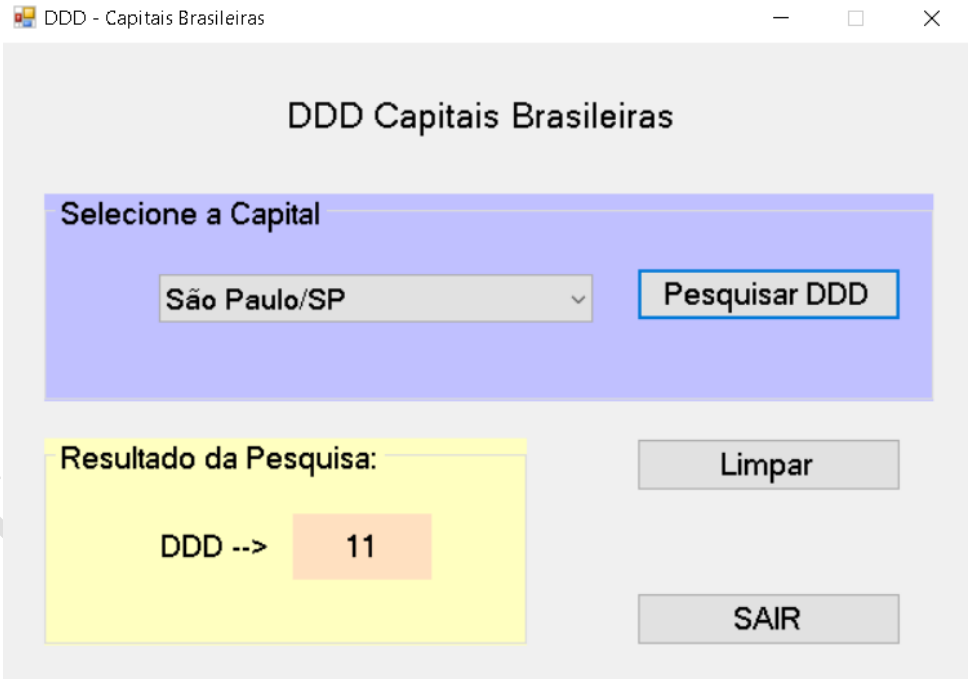
Estrutura condicional: Switch/Case

Exe016 → A partir da seleção de uma capital brasileira, o aplicativo deverá exibir o "DDD" correspondente.

Vamos a uma solução "preliminar" (e resumida com alguns poucos DDDs) utilizando o conhecimento que temos até o momento, com a estrutura "if...else":

DDD	CIDADE/ESTADO
11	São Paulo /SP
21	Rio de Janeiro / RJ
27	Vitória / ES
31	Belo Horizonte / MG
41	Curitiba / PR
48	Florianopolis / SC
51	Porto Alegre / RS
61	Brasilia / DF
62	Goiania / GO
63	Palmas / TO

INTERFACE GRÁFICA



A programação do botão "Pesquisar", poderá ser algo do tipo:

```
private void BtnPesquisar_Click(object sender, EventArgs e)
    {
      if (CboCapital.Text == "São Paulo/SP")
```

```
        {
            LblDDD.Text = "11";
        }
        else if (CboCapital.Text == "Rio de Janeiro/RJ")
        {
            LblDDD.Text = "21";
        }
        else if (CboCapital.Text == "Vitória/ES")
        {
            LblDDD.Text = "27";
        }
        //else if....
        //
        else if (CboCapital.Text == "Palmas/TO")
        {
            LblDDD.Text = "63";
        }

    }
```

Percebe-se que ao longo do bloco "If" um membro da condição que se repete. Estamos SEMPRE comparando a mesma informação "CboCapital.Text", com outros valores!!!

Para estes casos, a estrutura condicional "switch….case" é a mais indicada para a solução do problema.

IMPORTANTE: Retornaremos a este projeto ao estudarmos "Array".

Sintaxe:

```
switch(VARIAVEL)
{
   case VALOR:
        //Faz algo se VARIAVEL for igual ao VALOR
        break;
   default:   //(OPCIONAL)
        //Faz algo se VARIAVEL não for igual a nenhum CASE
        break;
}
```

IMPORTANTE: A instrução "switch" só pode ser utilizada em tipos como "**int" ou "string**". Com qualquer outro tipo (float, double...), deveremos utilizar a estrutura if!!!!

SOLUÇÃO UTILIZANDO A ESTRUTURA SWITCH....CASE

```
using System;
using System.Windows.Forms;

namespace Exe016
{
    public partial class FrmExe016 : Form
    {
        public FrmExe016()
        {
            InitializeComponent();
        }
```

```
private void FrmExe016_Load(object sender, EventArgs e)
{
    //OBS: Propriedade SORTED está TRUE
    //     Propriedade DropDownStyle = DropDownList

    CboCapital.Items.Add("São Paulo/SP");
    CboCapital.Items.Add("Rio de Janeiro/RJ");
    CboCapital.Items.Add("Vitória/ES");
    CboCapital.Items.Add("Belo Horizonte/MG");
    CboCapital.Items.Add("Curitiba/PR");
    CboCapital.Items.Add("Florianópolis/SC");
    CboCapital.Items.Add("Porto Alegre/RS");
    CboCapital.Items.Add("Brasilia/DF");
    CboCapital.Items.Add("Goiania/GO");
    CboCapital.Items.Add("Palmas/TO");

    CboCapital.SelectedIndex = 0;
}

private void BtnPesquisar_Click(object sender, EventArgs e)
{
    string cidade = CboCapital.Text;

    switch (cidade)
    {
        case "São Paulo/SP":
            LblDDD.Text = "11";
            break;
        case "Rio de Janeiro/RJ":
            LblDDD.Text = "21";
            break;
        case "Vitória/ES":
            LblDDD.Text = "27";
            break;
        case "Belo Horizonte/MG":
            LblDDD.Text = "31";
            break;
        case "Curitiba/PR":
            LblDDD.Text = "41";
            break;
        case "Florianópolis/SC":
            LblDDD.Text = "48";
            break;
        case "Porto Alegre/RS":
            LblDDD.Text = "51";
            break;
        case "Brasilia/DF":
            LblDDD.Text = "61";
            break;
        case "Goiania/GO":
            LblDDD.Text = "62";
            break;
        default:
            LblDDD.Text = "63";
            break;
    }
}

private void BtnSair_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

```
        private void BtnLimpar_Click(object sender, EventArgs e)
        {
            LblDDD.Text = "";
            CboCapital.SelectedIndex = 0;
            BtnPesquisar.Focus();
        }
    }
}
```

Estruturas de repetição

Também conhecidas como estruturas de looping, permitem a execução de uma (ou várias) instrução repetidamente até que uma condição seja verdadeira.

1 - While → Sintaxe da estrutura de repetição "**While"**:

```
while (condição)
{
  comandos;
}
```

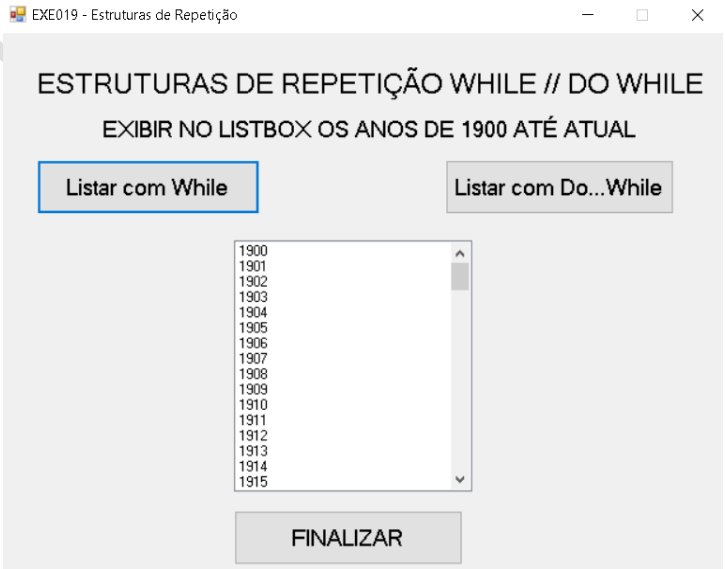
IMPORTANTE: O teste "condicional" é efetuado ANTES da execução do(s) comando(s).

2 - "Do While": Sintaxe da estrutura de repetição "**do While"**:

```
do
{
      comandos;
}
while (condição);
```

IMPORTANTE: Ao contrário da estrutura "while", o teste de "condição" é efetuado **após** a execução dos comandos. O que significa que o bloco de comandos será executado pelo menos uma vez.

Veremos no exemplo abaixo, o mesmo tipo de problema, porém, com duas soluções diferentes:



A solução apresenta os recursos da estrutura de repetição, afinal, seria uma tarefa bem árdua fazer esta programação exibindo os anos “um por um”...

Veja a programação:

SOLUÇÃO

```
using System;
using System.Windows.Forms;
namespace Exe019

{
   public partial class Frm019 : Form
   {
      public Frm019()
      {
         InitializeComponent();
      }

      private void BtnSair_Click(object sender, EventArgs e)
      {
         Application.Exit();
      }

      private void BtnListarWhile_Click(object sender, EventArgs e)
      {
         LstAnos.Items.Clear();
         int anoAtual = 0;
         anoAtual = Convert.ToInt16(DateTime.Now.ToString("yyyy"));
         int anoInicial = 1900;
         while (anoInicial <= anoAtual)
         {
            LstAnos.Items.Add(anoInicial);
            anoInicial++;   //incrementa 1 no anoInicial
         }
      }

      private void BtnListarDoWhile_Click(object sender, EventArgs e)
      {
         LstAnos.Items.Clear();
         int anoAtual = 0;
        anoAtual = Convert.ToInt16(DateTime.Now.ToString("yyyy"));
         int anoInicial = 1900;
         do
         {
            LstAnos.Items.Add(anoInicial);
            anoInicial++;   //incrementa 1 no anoInicial
         }
         while (anoInicial <= anoAtual);
      }
   }
}
```

Os valores exibidos no "ListBox", com as duas propostas, serão os mesmos, mas, para poder perceber a diferença entre uma estrutura e outra, vamos "forçar" e "fixar" o valor inicial da variável "anoAtual" com o número 1800. Faça a alteração e execute o problema.

Perceberemos que ao clicar no botão "Exibir com While" nenhum valor será exibido no ListBox, porém, quando clicamos no botão "Exibir com Do...While", o aplicativo exibirá 1900. Por que isto ocorre?

Dúvida: O que ocorrerá e a instrução "anoInicial++" não for utilizada?? Faça o teste.

3 - Estrutura de repetição "For": A sintaxe da estrutura "For" possui três parâmetros que devem ser fornecidos para que funcione corretamente. O bloco de comandos será executado até que a condição proposta seja satisfeita. Sintaxe da estrutura:

```
For (<inicio>;<condição>;incremento>)
{
        comandos;
}
```

Vamos utilizar o mesmo projeto anterior, agora com mais esta solução, vamos incluir um novo botão: "Listar com For". Veja a programação:

```
private void BtnListarFor_Click(object sender, EventArgs e)
    {
      LstAnos.Items.Clear();

      int anoAtual = 0;
      anoAtual = Convert.ToInt16(DateTime.Now.ToString("yyyy"));

      for (int anoInicial = 1900; anoInicial <= anoAtual; anoInicial++)
      {
        LstAnos.Items.Add(anoInicial);
      }

   }
```

A estrutura "For" apresenta uma solução mais enxuta, pois incorpora no próprio cabeçalho:

- a declaração da variável e o valor inicial desta variável (int anoInicial=1900);

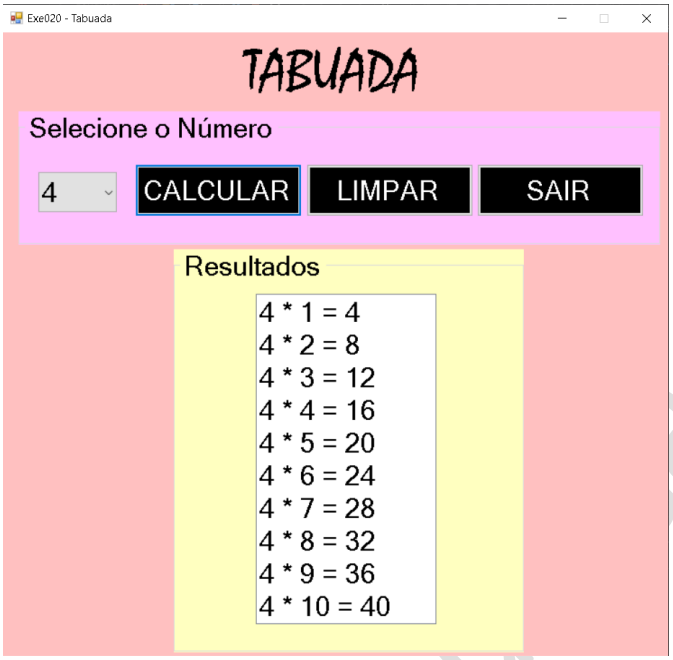
- a condição em que o "loop" deverá ser exectado (anoInicial <= anoAtual) e

- o incremento na variável "anoInicial".

Mas precisamos ficar atentos pois esta estrutura não servirá para o caso em que a condição final não for uma expressão numérica.... Veremos mais exemplos!!

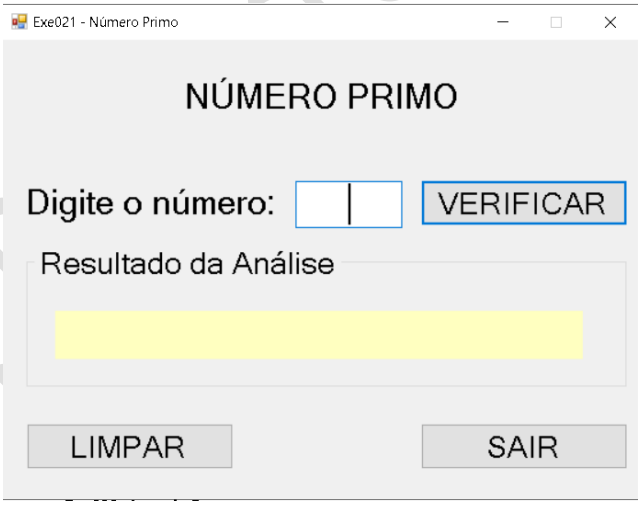
Exercício proposto Exe020: Tabuada

INTERFACE



Exercício proposto Exe021: Números primos

INTERFACE



Definição: Número "primo" é o número natural (inteiro e positivo) que pode ser dividido por apenas dois fatores: pelo **número** um e por ele mesmo. É importante perceber que o **número** um não é considerado um **número primo**, porque ele é divisível apenas por ele mesmo. Por outro lado, o **número** dois é o único **número primo** que também é um **número** par. Portanto, todos os demais números **primos** dessa sequência numérica considerada infinita serão obrigatoriamente ímpares. E o **número** 0, é **primo**? Utilizando a mesma definição, a resposta continua a ser não. Já que um **número primo** é divisível por ele próprio e **zero** não pode ser dividido por **zero**, já que é uma indeterminação. (fonte: https://www.coc.com.br).

Alguns números primos para teste: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, ...