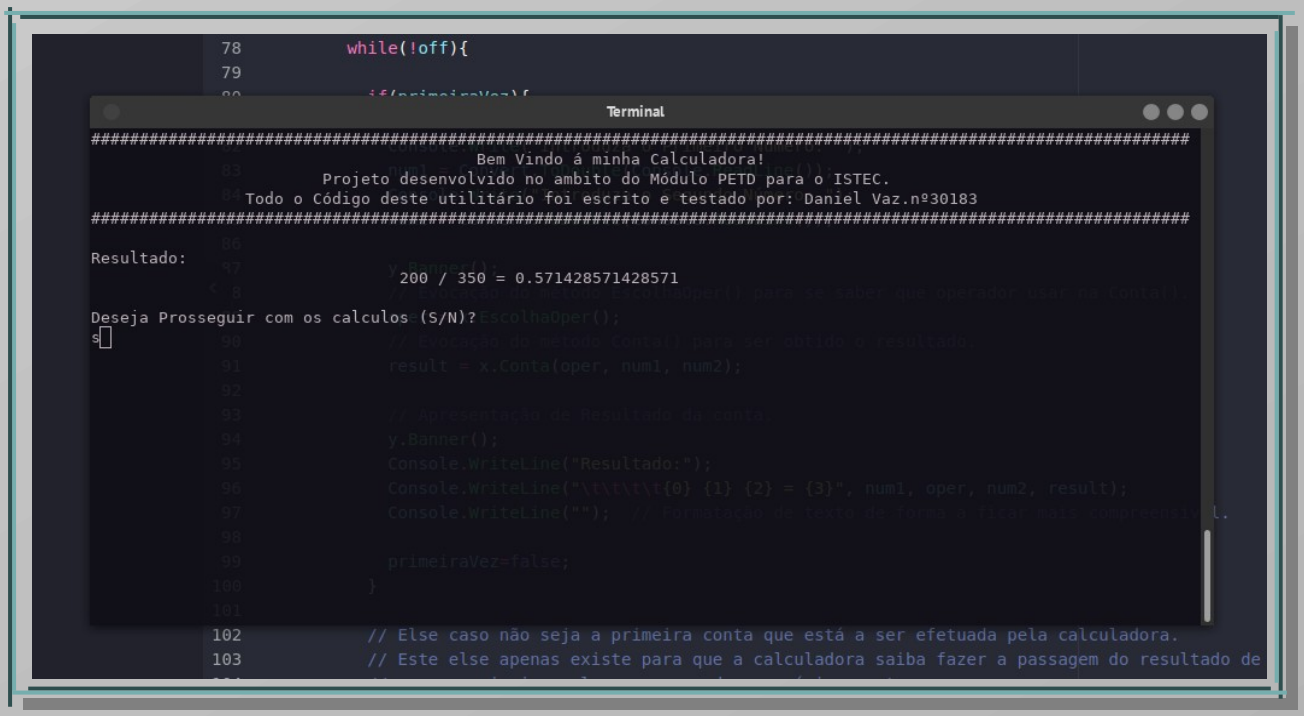
Relatório

Calculadora sem Interface Gráfica escrita em C#



Trabalho realizado por **Daniel Vaz**
nº30183-2A-RSI

Introdução

Este trabalho surge a pedido do Professor Sandro Ferreira no módulo de **P**rogramação **E**struturada e **T**ipo de **D**ados, do *Instituto Superior de Tecnologias Avançadas*.

Com este trabalho procurava-se ser documentada e criada uma Calculadora básica com recurso há linguagem de programação C#. Esta calculadora de forma a ser aprovada teria de cumprir os seguintes requisitos:

- Deverá ser capaz de executar as funções básicas de uma calculadora (Soma, Subtração, Divisão e Multiplicação);
- As operações deverão poder ser realizadas com um conjunto de operandos indefinidos, ou seja, podendo ser aplicadas operações ao resultado obtido da anterior operação;
- Tem de ser escolhida uma das seguintes *FrameWorks*: Console Apps, WPF, Xamarin.Forms, GDK;

Com os requisitos em mente, neste trabalho será documentado o meu processo lógico em torno da aplicação construida. Note-se que decidi desde partida construir a mesma através da *FrameWork* "*Console Apps*" , ou seja, será uma calculadora sem qualquer interface gráfico, sendo que todo o seu manuseamento será efetuado com recurso a um terminal de comandos.

De destacar também que esta aplicação foi construida num sistema de base *Fedora28*, onde todo o código foi escrito no *IDE Atom*, e o programa foi compilado e executado com ajuda dos utilitários provenientes do *Mono-Project*.

Source Code

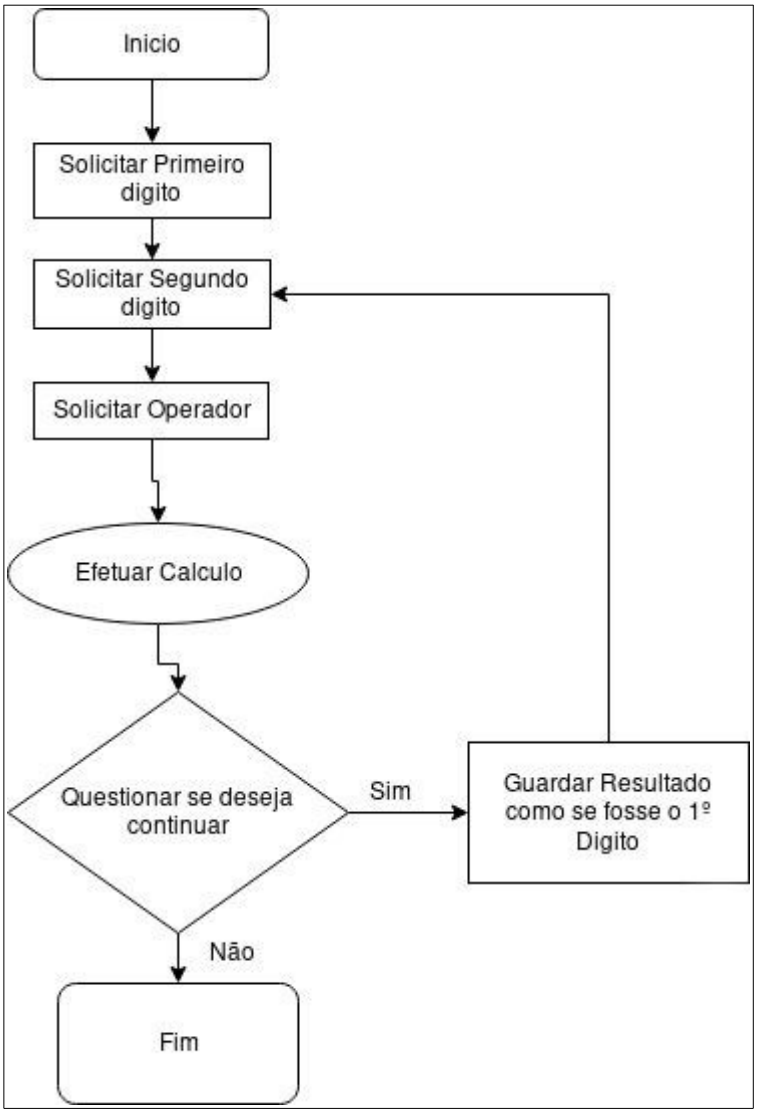
Todo o código que existe neste projeto foi desenhado, construido e testado por mim, tomando recurso da documentação oficial tanto da linguagem C# como da biblioteca Mono.

Poderá ser consultado o código fonte do projeto no mesmo *.zip* que continha este documento, no ficheiro de nome "***Calculadora.cs***" , contudo e por questões de redundância também poderá ser consultado através do seguinte link.

Planeamento e Estrutura Lógica do Programa

Para serem respeitados todos os requisitos impostos no projeto, como ponto de partida decidi que o mesmo deveria ter algum tipo de *loop* infinito, de modo a poder de forma continua solicitar valores ao utilizador e ir tomando ações sobre os mesmos.

Devido a ter algumas linhas de orientação contudo genéricas, decidi autonomamente que o programa ia ter a seguinte estrutura exemplar:



Com este esquema em mente, podemos abordar alguns excertos de código para perceber melhor como é que foi aplicado este funcionamento e lógica em *CSharp*.

Banner de Apresentação:

Começando pelo inicio, no programa por questões estéticas foi criado um *Banner,* que providenciava ao utilizador algumas informações básicas sobre o mesmo. Este *banner* foi posto em prática com recurso a uma classe própria que foi classificada por "***Apresentação***" assim, uma vez instanciada esta classe na nossa classe "*Main*" poderíamos evocar o método nela existente as vezes que quiséssemos sem estar a rescrever o código sistematicamente.

Apesar de ser uma funcionalidade apenas estética permite organizar melhor a informação e facilita o foco do utilizador para o programa, mesmo tratando-se de um sem qualquer interface gráfico:

```
  // Class responsável pela limpeza do ecrã e colocação da Banner de Apresentação.
  public class Apresentaçao{
    public void Banner(){
      Console.Clear(); //Limpar Ecrã.

Console.WriteLine("##############################################################################
##########################################");
      Console.WriteLine("\t\t\t\t\tBem Vindo á minha Calculadora!");
      Console.WriteLine("\t\t\tProjeto desenvolvido no âmbito do Módulo PETD para o
ISTEC.");
      Console.WriteLine("\t\tTodo o Código deste utilitário foi escrito e testado por:
Daniel Vaz.nº30183");

Console.WriteLine("##############################################################################
##########################################");
      Console.WriteLine("");
    }
  }
```

(Outra classe e métodos foram criados contudo a sua explicação será dada mais a frente)

Corpo do programa:

No inicio do "*Main*" após a criação de algumas variáveis globais, decidi iniciar o meu *loop* infinito que iria-me fazer *N* número de operações, sendo que dentro desse *loop* tinha de se ter algum mecanismo que me permiti-se identificar se a operação que estava a ser efetuada seria o primeiro calculo ou não, isto pois caso fosse o primeiro calculo, seria necessário solicitar 2 números ao utilizador, contudo caso se trata-se de uma operação que deveria continuar tomando em conta o resultado da operação anterior, então, deveria ser guardado esse resultado e solicitado apenas 1 valor ao cliente juntamente com a nova operação pretendida.

```
        // Criação de variáveis globais.
        bool off=false;
        bool primeiraVez=true;
        double num1=0, num2=0, result=0;
        string oper="";
        //Instanciamento das Classes necessárias.
        Operaçoes x = new Operaçoes();
        Apresentaçao y = new Apresentaçao();

        // Limpeza de Ecrã e colocação da Banner.
        y.Banner();

        // Loop infinito responsável por obter e calcular valores até que seja dito em
contrário pelo próprio Utilizador.
        while(!off){

          if(primeiraVez){
            //Obtenção de valores.
            Console.Write("Introduza o Primeiro Número: ");
            num1 = Convert.ToDouble(Console.ReadLine());
            Console.Write("Introduza o Segundo Número: ");
            num2 = Convert.ToDouble(Console.ReadLine());

            y.Banner();
            // Evocação do método EscolhaOper() para se saber que operador usar na
Conta().
            oper = x.EscolhaOper();
            // Evocação do método Conta() para ser obtido o resultado.
            result = x.Conta(oper, num1, num2);

            // Apresentação de Resultado da conta.
            y.Banner();
            Console.WriteLine("Resultado:");
            Console.WriteLine("\t\t\t\t{0} {1} {2} = {3}", num1, oper, num2, result);
            Console.WriteLine("");  // Formatação de texto de forma a ficar mais
compreensivel.

            primeiraVez=false;
          }

          // Else caso não seja a primeira conta que está a ser efetuada pela
calculadora.
          // Este else apenas existe para que a calculadora saiba fazer a passagem do
resultado de uma conta feita anteriormente,
          // para o primeiro valor a ser usado na próxima conta.
          else{
            Console.WriteLine("Deseja Prosseguir com os calculos (S/N)?");
            string opc = Console.ReadLine();

            y.Banner();
            if (opc == "S" || opc == "s"){
              num1 = result; // Resultado da Conta anterior é agora o nosso primeiro
valor "num1". Não sendo assim novamente solicitado ao utilizador.
              oper = x.EscolhaOper();
              Console.Write("Introduza o Segundo Número: ");
              num2 = Convert.ToDouble(Console.ReadLine());

              result = x.Conta(oper, num1, num2);

              y.Banner();
              Console.WriteLine("Resultado:");
              Console.WriteLine("\t\t\t\t{0} {1} {2} = {3}", num1, oper, num2, result);
              Console.WriteLine("");  // Formatação de texto de forma a ficar mais
compreensivel.
            }
(...)
```

No texto acima destaco a vermelho as duas variáveis responsáveis por manter apenas valores booleanos, já que estas é que são as responsáveis por distinguir no programa em que "estado" é que o mesmo está, ou seja, se ainda está no *loop* infinito ou não e se estamos na nossa primeira operação ou não.

A Azul está identificado o instanciamento de 2 classes de forma a poder ser usado os métodos nelas contidos. Uma das classes já foi citado acima sendo esse o responsável pela limpeza do ecrã e colocação da *banner* inicial, a segunda é a classe de "***Operações***" onde nela estão contidos 2 métodos, o "***Conta()***" e o "***EscolhaOper()***", estes são responsáveis por calcular o resultado final e escolha\confirmação de um operador a ser usado. *(Estes métodos serão mais a frente revistos e explicados em maior detalhe)*

E finalmente no excerto principal a verde, vemos todo o manuseamento dos números necessários para as diferentes operações. Note-se então, que caso a nossa variável primeiraVez não seja alterada é solicitado ao user o num1 e o num2, contudo caso não seja já a primeira operação a ser executada e a variável primeiraVez já tenha sido modificada, o result anterior é passado para o num1 e é apenas solicitado o num2.

Fim do Loop:

Porem, de forma a terminar o programa, o utilizador tinha de ter uma maneira de sair deste loop infinito e para isso mesmo é que foi criada a condicionante marcada no excerto com a cor amarela. Esta condicionante é sempre apresentada ao utilizador após ser efetuado um determinado calculo, assim, o utilizador tem a liberdade para escolher se pretende continuar a efetuar operações com o número que já estava a trabalhar anteriormente, ou se pretende sair do programa. Eis a essa mesma condicionante na sua totalidade:

```
            if (opc == "S" || opc == "s"){
              num1 = result; // Resultado da Conta anterior é agora o nosso primeiro
valor "num1". Não sendo assim novamente solicitado ao utilizador.
              oper = x.EscolhaOper();
              Console.Write("Introduza o Segundo Número: ");
              num2 = Convert.ToDouble(Console.ReadLine());

              result = x.Conta(oper, num1, num2);

              y.Banner();
              Console.WriteLine("Resultado:");
              Console.WriteLine("\t\t\t\t{0} {1} {2} = {3}", num1, oper, num2, result);
              Console.WriteLine("");  // Formatação de texto de forma a ficar mais
compreensivel.
```

```
            }

            else if (opc == "N" || opc == "n"){
              Console.WriteLine("");
              Console.WriteLine("\t\t\tObrigado por ter usado a Calculadora!");
              off = true; // Alteração de boolean para se sair do while loop infinito.
              Console.ReadLine();
            }
            else{
              Console.WriteLine("");
              Console.WriteLine("\t\tNão foi escolhida uma opção válida! A calculadora
irá terminar. ");
              off = true;
              Console.ReadLine();
            }
```

E tal como pode ser verificado caso o utilizador escolha "*N*" ou introduza qualquer outro valor se não um daqueles indicados, o programa altera o valor da nossa já anteriormente referida variável "off" e assim sai do *loop* infinito e termina o programa com a devida mensagem de despedida.

Métodos:

Para facilitar a leitura do código fonte do projeto e para reduzir as vezes que teria de rescrever o mesmo código, foi criada uma classe de nome "***Operações***", aqui residem 2 métodos essenciais ao funcionamento do meu programa: o "***Conta()***" e o "***EscolhaOper()***", sendo que podem ser analisadas mais abaixo:

```
(...)
  // Class que armazena métodos relacionados com Operações e Contas.
  public class Operaçoes {

    public string EscolhaOper(){
      string oper="";
      while(oper==""){
        Console.WriteLine("\t\t\tQue Operação é que pretende efetuar ?");
        Console.WriteLine("\t\t[ +(soma) ; -(subtração) ; *(multiplicação) ;
/(divisão) ]");
        oper = Console.ReadLine();
        if(oper == "+" || oper == "-" || oper == "*" || oper == "/"){
          break;
        }
        else{
          Console.WriteLine("Introduza um operador válido! (Tem de introduzir o seu
respetivo simbolo)");
          oper="";
        }
      }
      return oper;
    } // Fim do metodo EscolhaOper().


    public double Conta(string oper, double num1, double num2){
      double result=0;
```

```
      switch(oper){
        case "+":
          result=num1+num2;
          break;
        case "-":
          result=num1-num2;
          break;
        case "*":
          result=num1*num2;
          break;
        case "/":
          result=num1/num2;
          break;
      }
      return result;
    } // Fim do metodo Conta().
  } //Fim da Class Operaçoes.
(...)
```

O método "*EscolhaOper()*" é bastante simples, apenas é responsável por solicitar a escolha de um operador ao cliente e confirmar se o *input* fornecido é válido ou não, sendo que caso o valor for válido será retornado o operador escolhido. Usando este método podemos armazenar o operador numa variável *string*, aqui classificada como "oper".

O método "*Conta*" já aceita como parâmetros todos os valores necessários para se fazer uma conta, ou seja, os números "num1" e "num2", e a operação que será efetuada "oper". Enviando estes parâmetros ao nosso método ele efetuará um *switch case* de forma a distinguir qual operador é que foi fornecido e pendente esse operador, a conta a ser efetuada entre os 2 números indicados.

Conclusão

Apesar de não ser perfeito, o programa aqui demonstrado cumpre os requisitos impostos inicialmente e eleva a fasquia do desafio estando preparado para corrigir alguns possíveis erros por parte do utilizador, tornando-o assim numa solução mais robusta e eficiente. Para mais, graças a estrutura criada com os diferentes métodos, damos liberdade ao programa para este ser facilmente expansível, podendo ser adicionado sem receio novas operações e ou mecanismos de controlo do utilizador (por exemplo um menu de escolha de ações mais completo).

Espera-se que associado com esta documentação se tenha uma fácil abordagem ao condigo fonte da calculadora, permitindo assim a sua total interpretação e análise.

Obrigado pela sua leitura.