Universidade Estadual de Maringá
Departamento de Informática
Curso: Informática
Disciplina: Projeto e Análise de Algoritmos
Professor: Rodrigo Calvo

Projeto e Análise de Algoritmos
Trabalho II

Integrante	RA
Douglas Mezuraro	95676
Ronnie Petterson Bueno	91938

Arquivos de código contidos no trabalho

- ***Pascal.Triangle**.dpr*: Arquivo do projeto Delphi
- ***Impl.PascalTriangle**.pas*: Arquivo com a implementação dos algoritmo de triângulo de Pascal
- ***Impl.Matrix**.pas*: Arquivo que contém a definição do tipo matriz e algoritmo para inicialização da mesma
- ***Form.Main**.pas*: Arquivo que é a interface gráfica utilizada para facilitar a interação do usuário
- ***Form.Main**.dfm*: Arquivo que o delphi utiliza para montar o formulário em tempo de execução

Outros arquivos arquivos com extensões que não sejam *.dpr*, *.pas* e *.dfm* são gerados e utilizados pelo *Delphi* para eventuais configurações adicionais do projeto.

Informações iniciais

A implementação foi feita em *object-pascal* (*Delphi*). O saída da execução do trabalho é mostrado em uma tabela no próprio formulário.

Instruções de execução

1. Abrir o arquivo bin/*Pascal.Triangle.exe*
2. Clicar em calcular para ver a saída

Implementação do algoritmo

Foi implementado uma classe chamada *TPascalTriangle* onde possui os seguintes métodos:

```
TPascalTriangle = class
private
  class function Pascal(const Row, Column: Integer; const Matrix: TMatrix): Integer; overload;
public
  class function Pascal(const Size: Integer): TMatrix; overload;
end;
```

O método público recebe como parâmetro o tamanho do triângulo de pascal e retorna um tipo chamado *TMatrix* que é um array multidimensional de inteiros.

Inicialmente é chamado o método *InitializeMatrix()* da classe *TMatrix* que dimensiona a matriz e inicializa todas as suas posições com uma constante passada como parâmetro:

```
function InitializeMatrix(const Size: Integer; const Value: Integer = UndefinedValue): TMatrix;
```

.

Após isso é iterado sobre todas as posições da matriz fazendo cada posição receber o valor da chamada do método privado *Pascal()* que por sua vez retorna o valor de cada posição da matriz.

```
class function Pascal(const Row, Column: Integer; const Matrix: TMatrix): Integer; overload;
```

Esse método recebe como parâmetros a linha e coluna que irá ser calculada e a matriz que irá ser o resultado do método público *Pascal()*. É sobre essa matriz que é aplicado a programação dinâmica onde no seguinte trecho de código:

```
if Matrix[Row][Column] <> UndefinedValue then
begin
  Result := Matrix[Row][Column];
  Exit;
end;
```

Evita que recursões sobre valores já calculados antes sejam feitas. Então a lógica seria: se já calculou para esta linha e coluna retorna o valor anteriormente calculado, senão, calcula, retorna e armazena o resultado na matriz para que futuramente possa ser consultado.