# Centro Universitário FEI

PÓS GRADUAÇÃO – ENGENHARIA ELÉTRICA

MESTRADO – IAAA

INTELIGÊNCIA ARTIFICIAL APLICADA

À AUTOMAÇÃO E ROBÓTICA

## Programação Científica – PEL 216

Prof. Dr. Reinaldo Augusto da Costa Bianchi

## INTEGRAÇÃO NUMÉRICA

"Cálculo Numéricos"

FLÁVIO INFANTI
Matrícula: 118.310-2

AULA 05 – 13/08/2019

# 1 OBJETO

Dar continuidade ao aprendizado da disciplina de Programação Científica e se aprofundar nas técnicas referentes à Programação Orientada a Objeto C++, com a realização de exercícios usando Integração Numérica, consolidando desta forma, os conhecimentos adquiridos durante aula realizada em 13/08/2019, com a implementação de um algoritmo para solucionar a integral de três equações.

# 2 INTRODUÇÃO

Métodos de integração numérica são processos que realizam a aproximação de integrais definidas utilizando aproximações matemáticas aplicadas em um intervalo desejado, retornando desta forma, valores muito próximos do que seria obtido no calculo numérico.

Assim sendo, existem dois motivos para procurarmos uma aproximação numérica para a integral da função $f(x)$, primeiramente quando não se conhece a função e $f(x)$ é conhecida mas é muito complexa, o que dificulta a determinação de sua primitiva.

A seguir apresentamos alguns métodos utilizados para determinar a integral $f(x)$ e aproximar integrais numericamente, sendo as mais comuns:

## 2.1 Regra do Ponto Médio ou dos Retângulos.

A regra do ponto médio consiste em subdividirmos o intervalo $[a, b]$ em "$n$" intervalos iguais que servirão para as bases dos retângulos a serem construídos e a soma das áreas será uma aproximação da integral definida da função $f(x)$ no intervalo $[a, b]$.

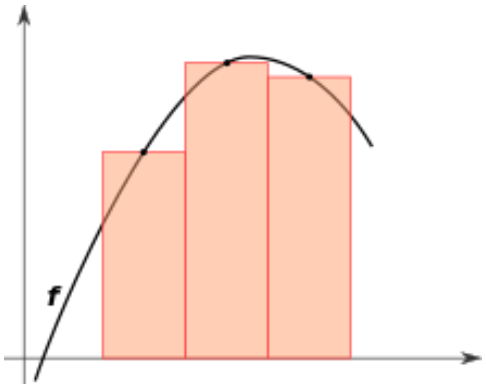
Essa função realiza a aproximação pela área do retângulo sobre a curva através de um polinômio de grau zero (constante).

Para realizar a integral aproximada de um intervalo usaremos a fórmula de Newton-Cotes e sua representação gráfica que para a regra do trapézio é dada por:

Fórmula 1- Newton-Cotes para regra do ponto médio

$$\int_a^b f(x)dx \simeq (b-a)f\left(\frac{a+b}{2}\right)$$

Gráfico 1 - Aproximação pela regra do ponto médio




Fonte: PowerPoint Aula 5 – 13/08/2019 - Prof. Reinaldo Bianchi - Centro Universitário FEI

## 2.2 Regra dos Trapézios

A regra dos trapézios, também consiste em subdividirmos o intervalo $[a, b]$ em "$n$" intervalos iguais que servirão para as bases dos trapézios a serem construídos e a soma das áreas será uma aproximação da integral definida da função $f(x)$ no intervalo $[a, b]$.

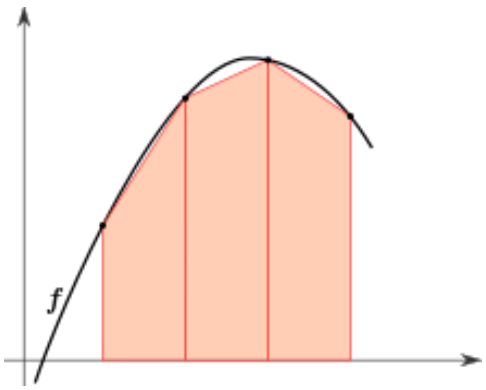
O nome do método vem do fato que a região entre o eixo x e a reta que liga os pontos sobre o gráfico da função nos extremos do intervalo forma um trapézio, desta forma, a aproximação é realizada sobre a curva através de um polinômio de grau um (linear) onde a integral é aproximada pela área do trapézio, que costuma apresentar melhores resultados em relação ao método do ponto médio, devido as áreas serem menores para cada intervalo [xi−1, xi].

Para realizar a integral aproximada de um intervalo usaremos a fórmula de Newton-Cotes e sua representação gráfica que para a regra do trapézio é dada por:

Fórmula 2 - Newton-Cotes para regra do trapézio

$$\int_a^b f(x)dx \simeq (b-a)\frac{f(a)+f(b)}{2}$$

Gráfico 2 - - Aproximação pela regra do trapézio




Fonte: PowerPoint Aula 5 – 13/08/2019 - Prof. Reinaldo Bianchi - Centro Universitário FEI

## 2.3 Regra de Simpson

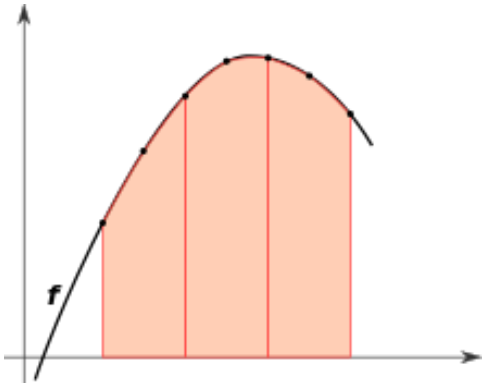
A aproximação pelo método de Simpson é mais elaborada e apresenta resultados mais significativos, devido ao fato de se usar dois subintervalos e aproximar a função f (x) por uma parábola neste intervalo, desta forma, a aproximação é realizada sobre a curva através de um polinômio de grau dois, portanto precisamos de três pontos do intervalo [a,b].

Para realizar a integral aproximada de um intervalo usaremos a fórmula de Newton-Cotes e sua representação gráfica que para a regra de Simpson é dada por:

**Fórmula 3 - Newton-Cotes para regra de Simpson**

$$\int_a^b f(x)dx \simeq (b-a)\frac{f(a)+4f(\frac{a+b}{2})+f(b)}{6}$$

**Gráfico 3 - Aproximação pela regra de Simpson**



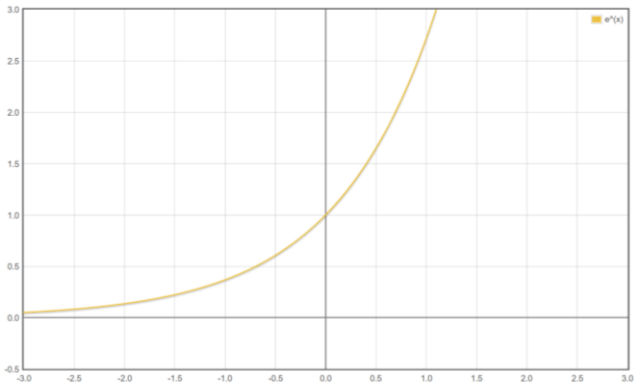

Fonte: PowerPoint Aula 5 – 13/08/2019 - Prof. Reinaldo Bianchi - Centro Universitário FEI

# 3 PROPOSTA DE IMPLEMENTAÇÃO.

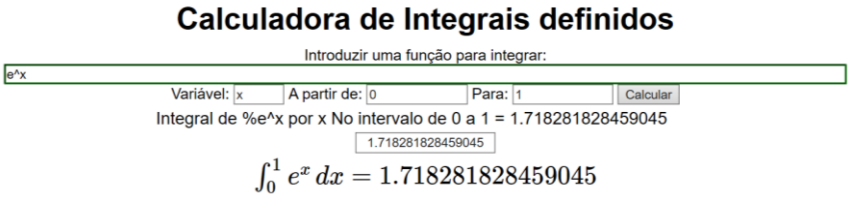
A proposta apresenta um algoritmo que aproxima a integral em três fases que são, decompor o intervalo de integração em sub-intervalos, realizar a integração aproximada da função de cada pedaço e somar os resultados numéricos obtidos.

Desta forma, o algoritmo foi implementado de forma a resolver as seguintes integrais:

## 3.1 Integral de $\int_0^1 e^x \, dx$

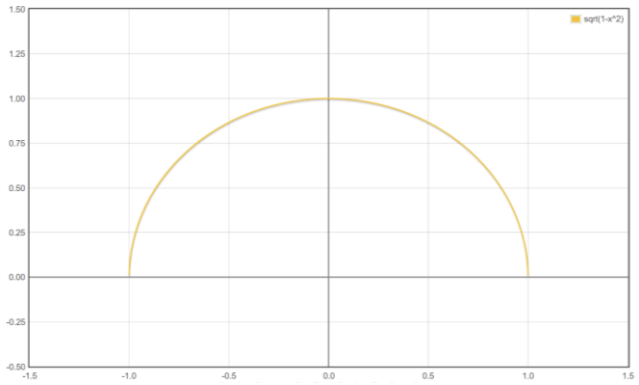



Fonte: Site Império dos Números (https://pt.numberempire.com/)

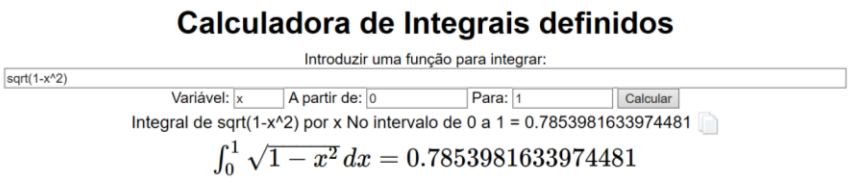



$$\int_0^1 e^x \, dx = 1.718281828459045$$

## 3.2 Integral de $\int_0^1 \sqrt{1-x^2} \, dx$

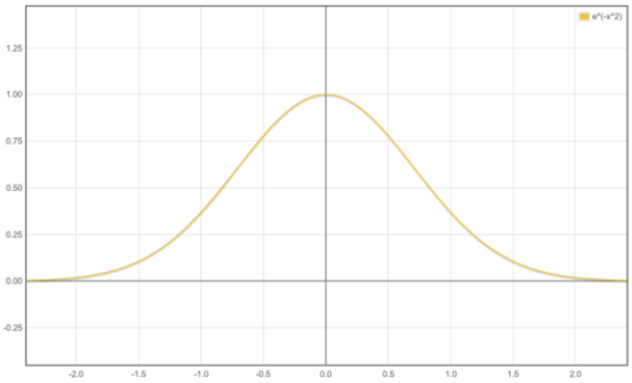



Fonte: Site Império dos Números (https://pt.numberempire.com/)




$$\int_0^1 \sqrt{1-x^2} \, dx = 0.7853981633974481$$

## 3.3 Integral de $\int_0^1 e^{-x^2}\,dx$




Fonte: Site Império dos Números (https://pt.numberempire.com/)

**Calculadora de Integrais definidos**

Introduzir uma função para integrar:

%e^(-x^2)

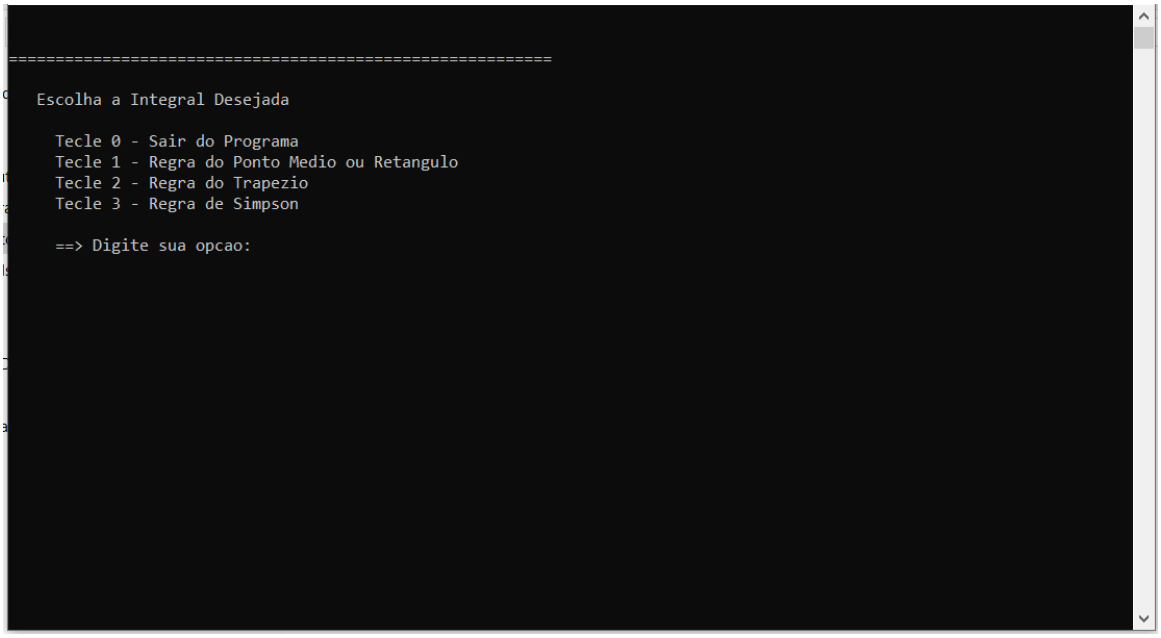
Variável: x A partir de: 0 Para: 1 Calcular

Integral de %e^(-x^2) por x No intervalo de 0 a 1 = 0.7468241328124271

$$\int_0^1 e^{-x^2}\,dx = 0.7468241328124271$$

Portanto, partiu-se da ideia de gerar um programa com um "Menu" (Figura 1) contendo uma opção para cada método de integração e dentro de cada opção realiza o calculo para as três integrais.

**Figura 1 - Tela Inicial do Programa**



# 4 RESULTADOS

Segue abaixo, telas com os resultados obtidas durante execução do algoritmo de integração.

Figura 2 - Tela com os Resultados para Regra Ponto Médio ou Retângulo

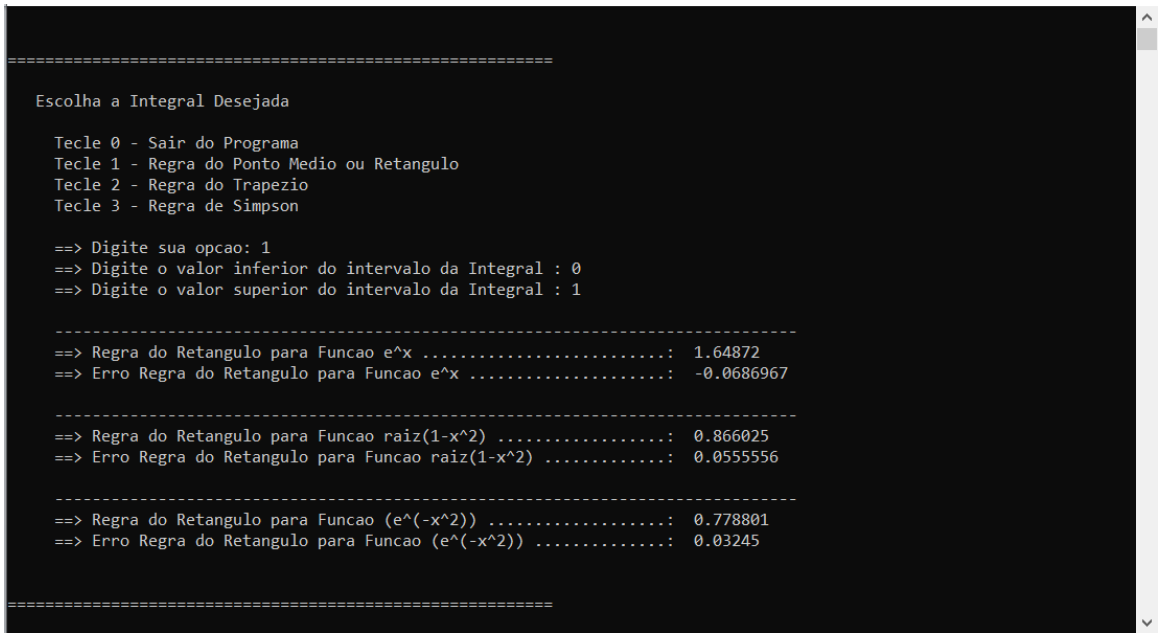



Figura 3 - Tela com os Resultados para Regra do Trapézio

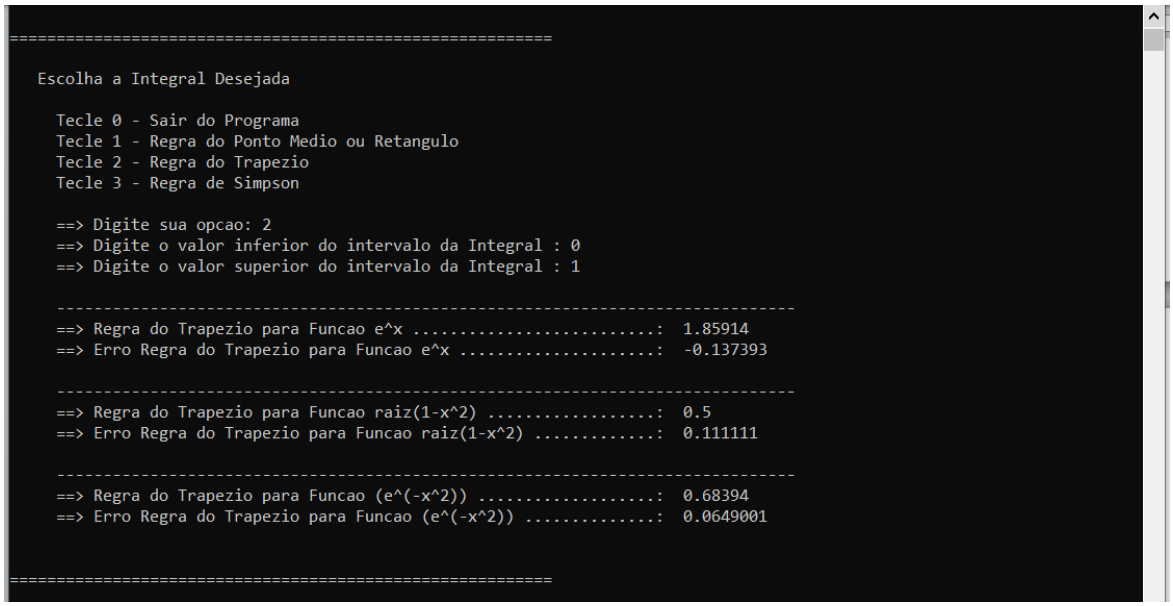
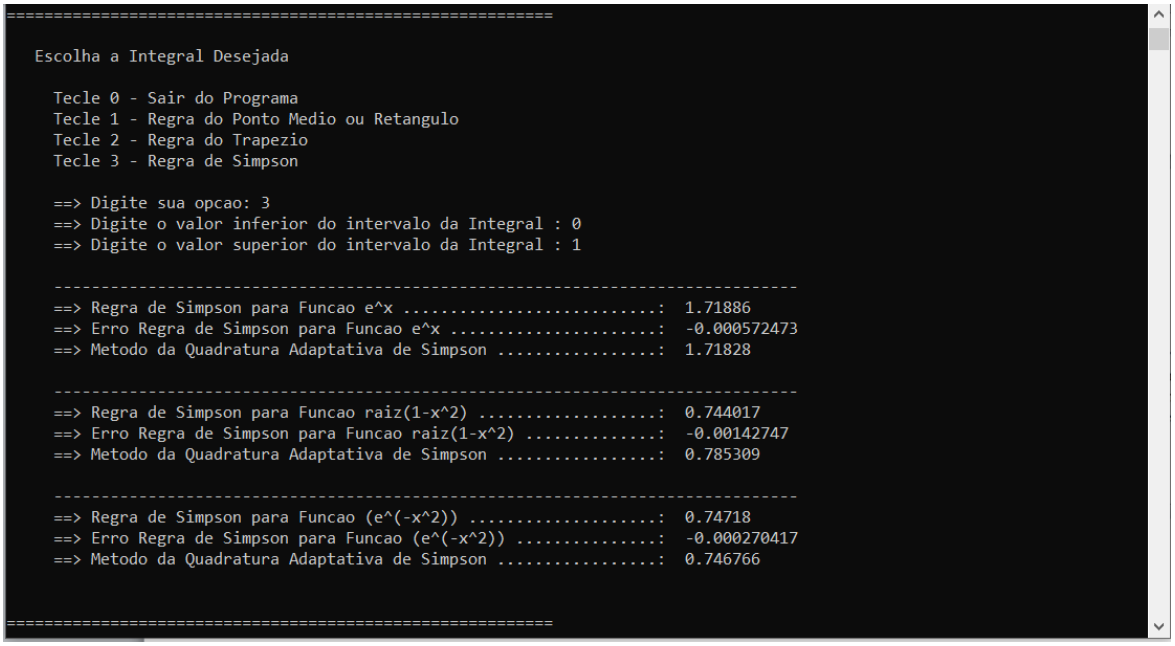


Figura 4 - Tela com os Resultados para Regra de Simpson




## 5 CONCLUSÕES

Como principal conclusão, podemos afirmar que o método é uma ferramenta importante para quando não conhecemos a função $f(x)$ e que quando conhecida ela é muito complexa dificultando desenvolver seu resultado.

No algoritmo proposto apresentam valores muito próximos do que seria obtido no calculo numérico.

E que a Regra de Simpson apresenta melhor resultado em relação às regras do retângulo e trapézio.

## 6 REFERÊNCIS BIBLIOGRAFIA


- VETTERLING, W.T.; PRESS W.H.; TEUKOLSKY, S.A.;FLANNERY, B.P. – Numerical Recipes, The Art of Scientific Computing, third edition, Cambridge, 2007.
- CHAPRA, S.C. Applied Numerical Methods –MCGrawHill,2011

# ANEXO – ALGORITMO DO PROGRAMA – “LISTA ENCADEADA”

```
//=== PEL216 - PROGRAMACAO CIENTIFICA
//=== Prof. Dr. Reinaldo Augusto da Costa Bianchi
//=== Aluno: Flavio Infanti nº 118.310-2
//=== AULA 05 - 13/08/2019
//===
//=== MÉTODOS NUMÉRICOS / INTEGRAÇÃO NUMÉRICA

//================================================================================
//==   BIBLIOTECAS
//================================================================================
#include <stdio.h>
#include <math.h>//necessária para usar as funções matemáticas
#include <iostream>
#include <cstdio>
#include <cstdlib>

using namespace std;

//================================================================================
//==   CALCULA O MODULO DE |X|
//================================================================================
float abs( float x )
{
	x = ( x * x ) / -(x);
	return x;
}


//================================================================================
//==   CALCULA O VALOR DA FUNÇÃO
//================================================================================
// função tipo 1 = (e^x)
float funcao_T1(float x)
{
	return exp(x);
}
// função tipo 2 = sqrt(1-x^2)
float funcao_T2(float x)
{
	return sqrt(1 - pow(x, 2));
}
// função tipo 3 = (e^(-x^2))
float funcao_T3(float x)
{
	return exp(-pow(x,2));
}


//================================================================================
//==   CALCULA O VALOR DA SEGUNDA DERIVADA
//================================================================================
// segunda derivada função tipo 1 = (e^x)
float derivada2_T1(float x)
{
	return exp(x);
}
// segunda derivada função tipo 2 = sqrt(1-x^2)
float derivada2_T2(float x)
{
	return -1 / pow((1 - pow(x, 2)), 3 / 2);
}
// segunda derivada função tipo 3 = (e^(-x^2))
float derivada2_T3(float x)
{
	return exp(-pow(x, 2))*(4*pow(x,2)-2);
}




//================================================================================
```

```
//==   CALCULA O VALOR DA QUARTA DERIVADA
//=============================================================================
// quarta derivada da função tipo 1 = (e^x)
float derivada4_T1(float x)
{
	return exp(x);
}
// quarta derivada da função tipo 2 = sqrt(1-x^2)
float derivada4_T2(float x)
{
	return -(12 * pow(x, 2) + 3) / pow((1 - pow(x, 2)), 7 / 2);
}
// quarta derivada da função tipo 3 = (e^(-x^2))
float derivada4_T3(float x)
{
	return 4 * exp(-pow(x, 2)) * (4 * pow(x, 4) - 12 * pow(x, 2) + 3);
}


//=============================================================================
//==   REGRA DO RETANGULO (PONTO MÉDIO)
//=============================================================================
// função tipo 1 = (e^x)
float retangulo_T1(float a, float b)
{
	return (b - a) * funcao_T1((a + b) / 2);
}
// função tipo 2 = sqrt(1-x^2)
float retangulo_T2(float a, float b)
{
	return (b - a) * funcao_T2((a + b) / 2);
}
// função tipo 3 = (e^(-x^2))
float retangulo_T3(float a, float b)
{
	return (b - a) * funcao_T3((a + b) / 2);
}


//=============================================================================
//==   ERRO PARA REGRA DO RETÂNGULO (PONTO MÉDIO)
//=============================================================================
// calcula o erro da função tipo 1 = (e^x)
float erroRetangulo_T1(float a, float b)
{
	return -((pow((b - a), 3) / 24) * derivada2_T1(a + ((b - a) / 2)));
}
// calcula o erro da função tipo 2 = sqrt(1-x^2)
float erroRetangulo_T2(float a, float b)
{
	return -((pow((b - a), 3) / 24) * derivada2_T2(a + ((b - a) / 2)));
}
// calcula o erro da função tipo 3 = (e^(-x^2))
float erroRetangulo_T3(float a, float b)
{
	return -((pow((b - a), 3) / 24) * derivada2_T3(a + ((b - a) / 2)));
}


//=============================================================================
//==   REGRA DO TRAPEZIO
//=============================================================================
// função tipo 1 = (e^x)
float trapezio_T1(float a, float b)
{
	return (b - a) * ((funcao_T1(a) + funcao_T1(b)) / 2);
}
// função tipo 2 = sqrt(1-x^2)
float trapezio_T2 (float a, float b)
{
	return (b - a) * ((funcao_T2(a) + funcao_T2(b)) / 2);
}
// função tipo 3 = (e^(-x^2))
float trapezio_T3 (float a, float b)
{
	return (b - a) * ((funcao_T3(a) + funcao_T3(b)) / 2);
}
//=============================================================================
//==   ERRO PARA REGRA DO TRAPEZIO
```

```
//================================================================================
// calcula o erro da função tipo 1 = (e^x)
float erroTrapezio_T1(float a, float b)
{
return -((pow((b - a), 3) / 12) * derivada2_T1(a + ((b - a) / 2)));
}
// calcula o erro da função tipo 2 = sqrt(1-x^2)
float erroTrapezio_T2(float a, float b)
{
	return -((pow((b - a), 3) / 12) * derivada2_T2(a + ((b - a) / 2)));
}
// calcula o erro da função tipo 3 = (e^(-x^2))
float erroTrapezio_T3(float a, float b)
{
	return -((pow((b - a), 3) / 12) * derivada2_T3(a + ((b - a) / 2)));
}


//================================================================================
// REGRA DE SIMPSON
//================================================================================
// função tipo 1 = (e^x)
float simpson_T1(float a, float b)
{
	return (b - a) * ((funcao_T1(a) + (4 * funcao_T1((a + b) / 2)) + funcao_T1(b)) / 6);
}
// função tipo 2 = sqrt(1-x^2)
float simpson_T2(float a, float b)
{
	return (b - a) * ((funcao_T2(a) + (4 * funcao_T2((a + b) / 2)) + funcao_T2(b)) / 6);
}
// função tipo 3 = (e^(-x^2))
float simpson_T3(float a, float b)
{
	return (b - a) * ((funcao_T3(a) + (4 * funcao_T3((a + b) / 2)) + funcao_T3(b)) / 6);
}


//================================================================================
//==   ERRO PARA REGRA DE SIMPSON
//================================================================================
// calcula o erro da função tipo 1 = (e^x)
float erroSimpson_T1(float a, float b)
{
	return -((pow((b - a), 5) / 2880) * derivada4_T1(a + ((b - a) / 2)));
}
// calcula o erro da função tipo 2 = sqrt(1-x^2)
float erroSimpson_T2(float a, float b)
{
	return -((pow((b - a), 5) / 2880) * derivada4_T2(a + ((b - a) / 2)));
}
// calcula o erro da função tipo 3 = (e^(-x^2))
float erroSimpson_T3(float a, float b)
{
	return -((pow((b - a), 5) / 2880) * derivada4_T3(a + ((b - a) / 2)));
}

//================================================================================
//==   METODO DA QUADRATURA ADAPTATIVA
//================================================================================
// funcao tipo 1 = (e^x)
float quadratura_adaptativa_T1(float a, float b, float erro)
{
	// calcula aproximação método de Simpson
	float Q_T1 = simpson_T1(a, b);
	float erroQ_T1 = erroSimpson_T1(a, b);
	// verifica se o erro atual da aproximação é menor que o erro desejado, se for maior executa
		if (abs(erroQ_T1) > erro)
		{
			// ponto médio entre os dois intervalos
			float pm = (a + b) / 2;
			//calcula aproximação do primeiro intervalor até o ponto médio
			float Q1_T1 = quadratura_adaptativa_T1(a, pm, erro);
			//calcula aproximação do ponto médio até o segundo intervalo
			float Q2_T1 = quadratura_adaptativa_T1(pm, b, erro);
			//soma aproximações parciais
			Q_T1 = Q1_T1 + Q2_T1;
		}
```

```
	return Q_T1;
}

// função tipo 2 = sqrt(1-x^2)
float quadratura_adaptativa_T2(float a, float b, float erro)
	{
	// calcula aproximação método de Simpson
	float Q_T2 = simpson_T2(a, b);
	float erroQ_T2 = erroSimpson_T2(a, b);
	// verifica se o erro atual da aproximação é menor que o erro desejado, se for maior executa
		if (abs(erroQ_T2) > erro)
		{
			// ponto médio entre os dois intervalos
			float pm = (a + b) / 2;
			//calcula aproximação do primeiro intervalor até o ponto médio
			float Q1_T2 = quadratura_adaptativa_T2(a, pm, erro);
			//calcula aproximação do ponto médio até o segundo intervalo
			float Q2_T2 = quadratura_adaptativa_T2(pm, b, erro);
			//soma aproximações parciais
			Q_T2 = Q1_T2 + Q2_T2;
		}
	return Q_T2;
}

// função tipo 3 = (e^(-x^2))
float quadratura_adaptativa_T3(float a, float b, float erro)
{
	// calcula aproximação método de Simpson
	float Q_T3 = simpson_T3(a, b);
	float erroQ_T3 = erroSimpson_T3(a, b);
	// verifica se o erro atual da aproximação é menor que o erro desejado, se for maior executa
		if (abs(erroQ_T3) > erro)
		{
			// ponto médio entre os dois intervalos
			float pm = (a + b) / 2;
			//calcula aproximação do primeiro intervalor até o ponto médio
			float Q1_T3 = quadratura_adaptativa_T3(a, pm, erro);
			//calcula aproximação do ponto médio até o segundo intervalo
			float Q2_T3 = quadratura_adaptativa_T3(pm, b, erro);
			//soma aproximações parciais
			Q_T3 = Q1_T3 + Q2_T3;
		}
	return Q_T3;
}

//=============================================================================
//==   DECLARA VARIÁVEIS
//=============================================================================
int a, b, opcao_Regra;
float Q_T1, Q_T2, Q_T3;


int main()
{

	while(1)
	{
	   printf ("\n\n=============================================================\n\n");
	   printf ("   Escolha a Integral Desejada          \n\n");
	   printf ("     Tecle 0 - Sair do Programa \n");
	   printf ("     Tecle 1 - Regra do Ponto Medio ou Retangulo\n");
	   printf ("     Tecle 2 - Regra do Trapezio\n");
	   printf ("     Tecle 3 - Regra de Simpson\n\n");
	   printf ("     ==> Digite sua opcao: ");
	   scanf("%i", &opcao_Regra);

		printf ("     ==> Digite o valor inferior do intervalo da Integral : ");
		scanf("%i", &a);

		printf ("     ==> Digite o valor superior do intervalo da Integral : ");
		scanf("%i", &b);

		// DEFINIÇÃO DO ERRO
		float erro = 0.0000001;


		float Q_T1 = quadratura_adaptativa_T1(a, b, erro);
		float Q_T2 = quadratura_adaptativa_T2(a, b, erro);
```

```
                float Q_T3 = quadratura_adaptativa_T3(a, b, erro);

            //=== ESCOLHE A REGRA DESEJADA
                switch(opcao_Regra)
                {
                case 0:
                    system("cls");
                        cout<<"  Ate em breve ..... TCHAU!!!."<<endl;
                    exit(1);
                    break;


        //************************************************************************
         //=== REGRA DO PONTO MEDIO OU RETANGULO
                case 1:
                {
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << " ==> Regra do Retangulo para Funcao e^x .....:  " << retangulo_T1(a, b) << endl;
                  cout << " ==> Erro Regra do Retangulo para Funcao e^x .....:  " << erroRetangulo_T1(a,b) << endl;
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << "==> Regra do Retangulo para Funcao raiz(1-x^2) .....:  " << retangulo_T2(a,b) << endl;
                  cout << "==> Erro Regra do Retangulo para Funcao raiz(1-x^2) .....:  " << erroRetangulo_T2 (a,b) << endl;
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << " ==> Regra do Retangulo para Funcao (e^(-x^2)) .....:  " << retangulo_T3(a,b) << endl;
                  cout << " ==> Erro Regra do Retangulo para Funcao (e^(-x^2)) .....:  " << erroRetangulo_T3(a,b) << endl;
                  break;
                }

                case 2:
                {
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << "    ==> Regra do Trapezio para Funcao e^x .....:  " << trapezio_T1(a, b) << endl;
                  cout << "    ==> Erro Regra do Trapezio para Funcao e^x .....:  " << erroTrapezio_T1(a,b) << endl;
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << "    ==> Regra do Trapezio para Funcao raiz(1-x^2) .....:  " << trapezio_T2(a,b) << endl;
                  cout << "    ==> Erro Regra do Trapezio para Funcao raiz(1-x^2) .....:  " << erroTrapezio_T2 (a,b) << endl;
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << "    ==> Regra do Trapezio para Funcao (e^(-x^2)) .....:  " << trapezio_T3(a,b) << endl;
                  cout << "    ==> Erro Regra do Trapezio para Funcao (e^(-x^2)) .....:  " << erroTrapezio_T3(a,b) << endl;
                  break;
                }

                case 3:
                {
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << "    ==> Regra de Simpson para Funcao e^x .....:  " << simpson_T1(a, b) << endl;
                  cout << "    ==> Erro Regra de Simpson para Funcao e^x .....:  " << erroSimpson_T1(a,b) << endl;
                  cout << "    ==> Metodo da Quadratura Adaptativa de Simpson .....:  " << Q_T1 << endl;
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << "    ==> Regra de Simpson para Funcao raiz(1-x^2) .....:  " << simpson_T2(a,b) << endl;
                  cout << "    ==> Erro Regra de Simpson para Funcao raiz(1-x^2) .....:  " << erroSimpson_T2(a,b) << endl;
                  cout << "    ==> Metodo da Quadratura Adaptativa de Simpson .....:  " << Q_T2 << endl;
                  cout << "\n    ----------------------------------------------------------------------------" << endl;
                  cout << "    ==> Regra de Simpson para Funcao (e^(-x^2)) .....:  " << simpson_T3(a,b) << endl;
                  cout << "    ==> Erro Regra de Simpson para Funcao (e^(-x^2)) .....:  " << erroSimpson_T3(a,b) << endl;
                  cout << "    ==> Metodo da Quadratura Adaptativa de Simpson .....:  " << Q_T3 << endl;
                  break;
                }
                }
                //cin.get();

        }
}
```