```c
/*
 Projeto de Cálculo Numérico
Prof. Alexandre Roma (IME-USP)

        IAG-USP
         2015

Fábio Oliveira - 7978417
Jessé Stenico - 9051932

Resolução de E.D.O. pelos métodos:
Euler Explícito,
Euler Implícito e
Runge-Kutta Clássico.

Os resultados obtidos serão comparados com a solução analítica conhecida previamente.

Caso de calibragem: Decaimento Radioativo

y'=-Lambda*y
y=exp(-Lambda*t)

onde "Lambda" depende do material radioativo em questão.
*/
#ifndef FUNCOES_H_INCLUDED
#define FUNCOES_H_INCLUDED

#define max_indice pow(2, TETO) + 1
static double ultimo_y_analitico;
static double ytf[3][TETO];
/*
Nesse caso,
ytf[0][j] - euler explicito
ytf[1][j] - euler implicito
ytf[2][j] - runge kutta
*/

void verifica_info(  ){
    printf  ("Intervalo: [%f, %f]\n"
             "Condicao de contorno: %f\n", t_MIN, t_MAX, y_contorno);
}

void inicializa_vetores(double n[ ], double h[ ]){

    int k;

    for(k=0; k<TETO; k++){
        n[k]=pow(2, k+1);
        h[k]=(t_MAX-t_MIN)/n[k];
    }
    for (k=0; k<TETO; k++){
        printf("Passos: %4.f DeltaT: %4f\n", n[k], h[k]);
    }
}

void resolve_analitico (double (*analitico_f)(double), double h[ ]){ /*SOLUÇÃO ANALÍTICA*/

    FILE *saida;
    double k;
    double max_passo=(t_MAX-t_MIN)/h[TETO-1];
    double y=y_contorno;
    double t=t_MIN;

    saida=fopen("y_analitico.txt", "w");

    fprintf (saida, "%.10f %.10f\n", t, y);
```

```c
    for (k=1; k<=max_passo; k++){

        t=t_MIN+k*h[TETO-1];
        y=analitico_f(t);
        fprintf (saida, "%.10f %.10f\n", t, y);

    }
    ultimo_y_analitico=y;
    fclose(saida);
}

void teste_convergencia (FILE *saida, double ytf[][TETO], int metodo){

    int k;
    double a, b;

    a=ytf[metodo][0];
    for (k=1; k<TETO; k++){

        b=ytf[metodo][k];
        fprintf(saida, "%d %.10f\n", k, fabs( (a-ultimo_y_analitico) / (b-ultimo_y_analitico) ));
        a=b;

    }
}

void euler_explicito (double (*fxy)(double, double), double n[ ], double h[ ]){

    FILE *saida;
    FILE *convergencia;

    char arq[15];
    int k;
    double y, t;
    double passo;


    for (k=0; k<TETO; k++){

        sprintf(arq, "exp%d.txt", (int)pow(2, k+1)); /*UM ARQUIVO DIFERENTE PARA CADA QUANTIDADE DE n
PASSOS*/
        saida=fopen(arq, "w");

        y=y_contorno;
        t=t_MIN;
        fprintf (saida, "%.10f %.10f\n", t, y);
        for(passo=1; passo<=n[k]; passo++){

            t=t_MIN+passo*h[k];
            y=y+h[k]*fxy(t, y);
            fprintf (saida, "%.10f %.10f\n", t, y);

        }
        fclose(saida);

        ytf[0][k]=y; /*ARMAZENA VALORES DE y(tf) EM UM VETOR SEPARADO*/
    }                               /*ÚTIL PARA TESTES DE CONVERGENCIA*/


    convergencia=fopen("conv_exp.txt", "w");
    teste_convergencia(convergencia, ytf, 0);
    fclose(convergencia);
}

void euler_implicito (double (*fxy)(double, double), double (*df)(double, double), double n[ ], double h[
]){
```

```c

    FILE *saida;
    FILE *convergencia;

    char arq[15];

    double erro;
    int passo, k, iteracao;

    double y_implicito;
    double t;
    double chute_inicial=(t_MAX-t_MIN)/2.0;
    double y1, y2;

    for (k=0; k<TETO; k++){ /*PERCORRE TODO O VETOR DE INCREMENTOS*/
        sprintf(arq, "imp%d.txt", (int)pow(2,k+1));
        saida=fopen(arq, "w");

        y1=chute_inicial;
        y_implicito=y_contorno;
        t=t_MIN;
        fprintf (saida, "%.10f %.10f\n", t, y_implicito);

        for(passo=1; passo<=n[k]; passo++){

            t=t_MIN+passo*h[k];
            erro=1/EPSILON; /*VALOR SEGURAMENTE MAIOR QUE EPSILON QUALQUER QUE SEJA ELE*/
            iteracao=1;
            while (erro>EPSILON){ /*METODO DE NEWTON*/

                if(iteracao>MAXIMO_ITERACAO){
                    printf ("Newton-Raphson: Atingiu maximo de iteracoes determinado!\n");
                    break;
                }

                y2=y1-(y1-y_implicito-h[k]*fxy(t, y1))/(1-h[k]*df(t, y1));
                erro=fabs(y2-y1);
                y1=y2;
                iteracao++;
            }
            y_implicito=y1;

            fprintf (saida, "%.10f %.10f\n", t, y_implicito);
        }
        fclose(saida);
        ytf[1][k]=y_implicito;
    }

    convergencia=fopen("conv_imp.txt", "w");
    teste_convergencia(convergencia, ytf, 1);
    fclose(convergencia);

}

void runge_kutta (double (*fxy)(double, double), double n[ ], double h[ ]){

    FILE *saida;
    FILE *convergencia;

    char arq[15];

    int k;

    double passo;
    double K_1, K_2, K_3, K_4;
    double y_rk;
```

```
    double t;

    for (k=0; k<TETO; k++){
        sprintf(arq, "rk%d.txt", (int)pow(2,k+1));
        saida=fopen(arq, "w");

        y_rk=y_contorno;
        t=t_MIN;
        fprintf (saida, "%.10f %.10f\n", t, y_rk);

        for(passo=1; passo<=n[k]; passo++){

            K_1=fxy(t, y_rk);

            K_2=fxy(t+0.5*h[k], y_rk+(0.5*h[k]*K_1));

            K_3=fxy(t+0.5*h[k], y_rk+(0.5*h[k]*K_2));

            K_4=fxy(t+h[k],     y_rk+(h[k]*K_3));

            y_rk=y_rk+(1.0/6.0)*h[k]*(K_1+K_2+K_2+K_3+K_3+K_4);

            t=t_MIN+passo*h[k];

            fprintf (saida, "%.10f %.10f\n", t, y_rk);

        }
        fclose(saida);
        ytf[2][k]=y_rk;

    }

    convergencia=fopen("conv_rk.txt", "w");
    teste_convergencia(convergencia, ytf, 2);
    fclose(convergencia);
}

#endif
```