6
Organizar um código fonte, ler e codificar um diagrama de blocos estruturado

Programa de Aprendizagem

Crie um programa para ler os valores dos coeficientes de uma equação do 2º grau A, B e C. para ler A,B e C. Crie funções de leitura (nâo void) para cada valor. Se o usuário digitar Zero no valor de A, exiba uma mensagem"A não poderá ser zero!" e finalize o programa. Se o valor de **A** for positivo, chame uma função não void para calcular o valor de **DELTA** = B*B*4*A*C. Depois pergunte se **DELTA** não é negativo, se for então exiba a mensagem "Delta não poderá ser Negativo!" e finalize o programa. Se DELTA for positivo, então chame uma função não void para calcular X1=-B + RAIZ(DELTA)/2*A e outra função não void para calcular X2=-B-RAIZ(DELTA)/2*A. Para exibir os resultados, crie o VOID EXIBIR ( ), dentro do void exibir, calcule o valor do delta, calcule o valor de x1 e x2 é exiba os resultados.

Algoritmo Descritivo

ALGORITMO DESCRITIVO

1 Criar as vars Reais A, B, C, DELTA;

2 Declare funções de leitura: Ler_A( ), Ler_B( ) e Ler_C( ). (A deve ser diferentes de Zero)

3 Crie a função de cálculo cal_Delta( ) com argumentos; DELTA=B*B-4*A*C;

4 Criar as funções de cálculo cal_x1( ) e cal_x2( ) , X1=-B+raiz(DELTA)/2*A e X2 = -B - raiz(DELTA) / 2*A;

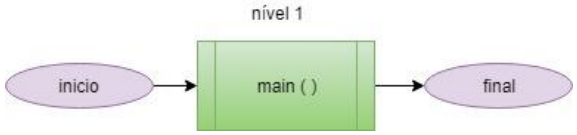
5 Crie o void exibir( ), dentro do void exibir, mostre A,B,C e DELTA; **SE** DELTA < 0, exiba "Impossível calcular X1 e X2, pois DELTA é negativo!", Finalize o Programa; **SENÃO** chame as funções cal_x1( ) para calcular X1 e cal_x2( ) para calcular X2, mostre X1 e X2;

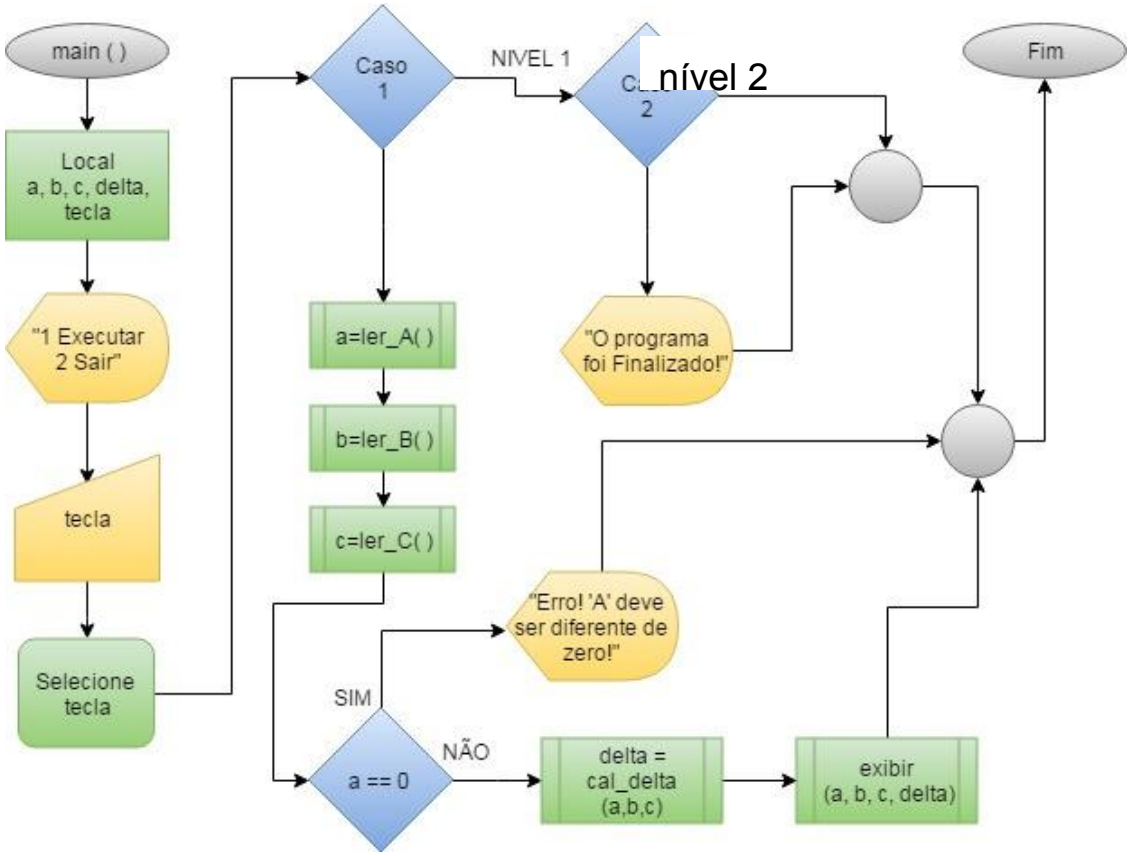
LÓGICA DE EXECUÇÃO NO SELECT CASE:

Menu 1 Executar: Chamar funções para ler A, B, C; Chamar função para calcular DELTA; Chame o void exibir ( );

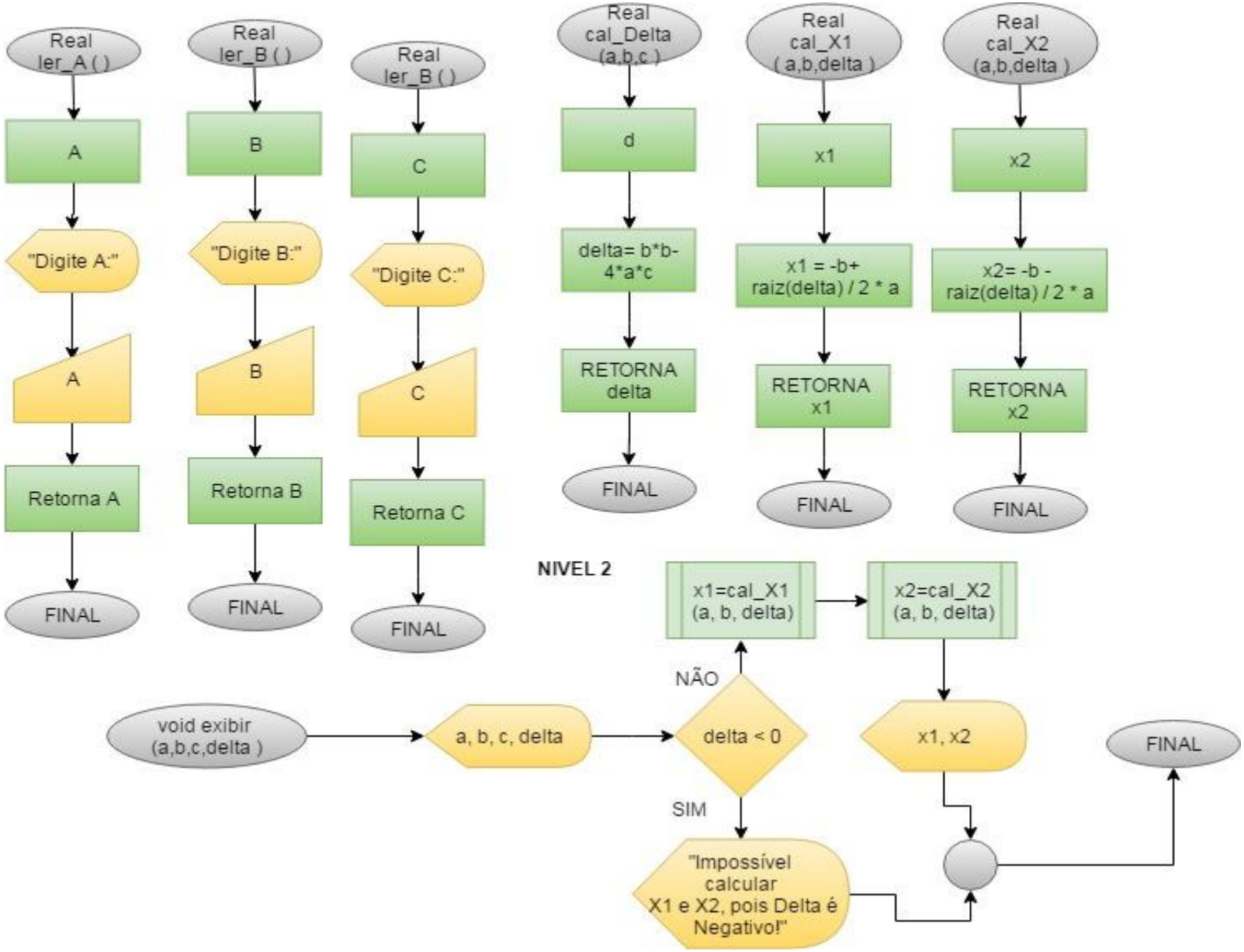
Menu 2 Sair: Finalize o Programa;

DIAGRAMA DE BLOCOS





Nível 3 (Sub Rotinas)



Código Fonte Estruturado
Funções de Leitura

As funções em um código em C++ devem ser codificadas logo após as diretivas do processador, exatamente na área do Escopo Global do Código, as primeiras funções devem ser as funções de leitura.

```
#include <iostream>
#include <cstdlib>
#include <math.h>
using namespace std;

double ler_A ( ) { double va;   cout << "Valor de A:";
cin >> va;    return va; }

double ler_B ( ) { double vb;   cout << "Valor de B:";
cin >> vb;    return vb; }

double ler_C ( ) { double vc;    cout << "Valor de C:";
cin >> vc;     return vc; }
```

/* continua no próximo slide … */

FUNÇÕES DE LEITURA

A função ler_A ( ) => Serve para ler e armazenar o valor de A

A função ler_B ( ) => Serve para ler e armazenar o valor de B

A função ler_C ( ) => Serve para ler e armazenar o valor de C

Funções de Cálculo

Após as funções de leitura, as próximas funções a serem codificadas, devem ser as funções de cálculo, neste programa as funções de cálculo são cal_Delta( ), cal_x1( ) e cal_x2( ).

```
double cal_Delta ( double a, double b, double c) {
 double d = pow(b,2) - 4 * a * c;
 return d; }

double cal_x1 ( double a, double b, double delta ) {
 double x = (-b + sqrt(delta))/(2*a);
 return x; }

double cal_x2 ( double a, double b, double delta ) {
 double x = (-b - sqrt(delta))/(2*a);
 return x; }
```

/* continua no próximo slide … */

FUNÇÕES DE CÁLCULO

A função cal_Delta ( ) => recebe os valores de a, b, e c, calcula e armazena o valor de delta, através do comando RETURN.

A função cal_x1 ( ) => recebe os valores de a, b e delta, calcula e armazena o valor de X1 e armazena através do comando RETURN.

A função cal_x2 ( ) =>recebe os valores de a, b e delta, calcula e armazena o valor de X1 e armazena através do comando RETURN.

Função de exibição/void de saída

Após as funções de cálculo, as funções que devem ser codificadas são as funções de saída, neste programa optou-se por criar um VOID de saída chamado exibir ( ) para exibir todos valores de saída.

```
void exibir ( double a, double b, double c, double
delta )
{
  cout << "\nValor de ....A:" << a;
   cout << "\nValor de ....B:" << b;
   cout << "\nValor de ....C:" << c;
   cout << "\nValor de Delta:" << delta;

if (delta >= 0)
    { double x1 = cal_x1(a,b,delta);
      double x2 = cal_x2(a,b,delta);
      cout << "\nValor de ....x1:" << x1;
      cout << "\nValor de ....x2:" << x2 << endl;}
 else   {  cout << "\nImpossível calcular x1 e
x2\nDelta é negativo!";}
system("pause"); }
```

/* continua no próximo slide … */

FUNÇÕES DE EXIBIÇÃO OU SAÍDA

A sub rotina void exibir ( ) => recebe e exibe os valores de a, b, c e delta, após calcular x1 e x2 chamando as funções cal_x1 ( ) e cal_x2 ( ), exibe x1 e x2. O tipo void não possui **RETURN** pois não armazena valores como as funções.

Função Principal Main ( )

Finalmente a função INT MAIN ( ) irá conter o menu principal de onde as funções poderão ser chamadas.

```
int main () {   setlocale(LC_ALL, "Portuguese");
double a, b, c, delta;     int tecla;
MENU:  /* marca um ponto no código para retornar aqui depois */
cout << "\nmenu\n1 Executar\n2 Finalizar\nitem:";      cin >> tecla;

switch(tecla)
{       case 1:  a = ler_A ( );    b = ler_B ( );     c = ler_C ( );

        if ( a == 0 )  {  cout << "\nErro!'A' deve ser diferente de zero!";
                          system("pause");   exit(0);  }

         else   {  delta = cal_Delta ( a, b, c );
                   exibir ( a, b, c, delta );          }  break;

 case 2:    cout << "\nO programa será finalizado!";   system ("pause");   exit ( 0 );   break;
}

goto MENU;   /* executa novamente a partir do ponto marcado */
return 0; }  /* fim do programa 4 */
```

Programa de Fixação

Diagrama Nível 1 main () com IF

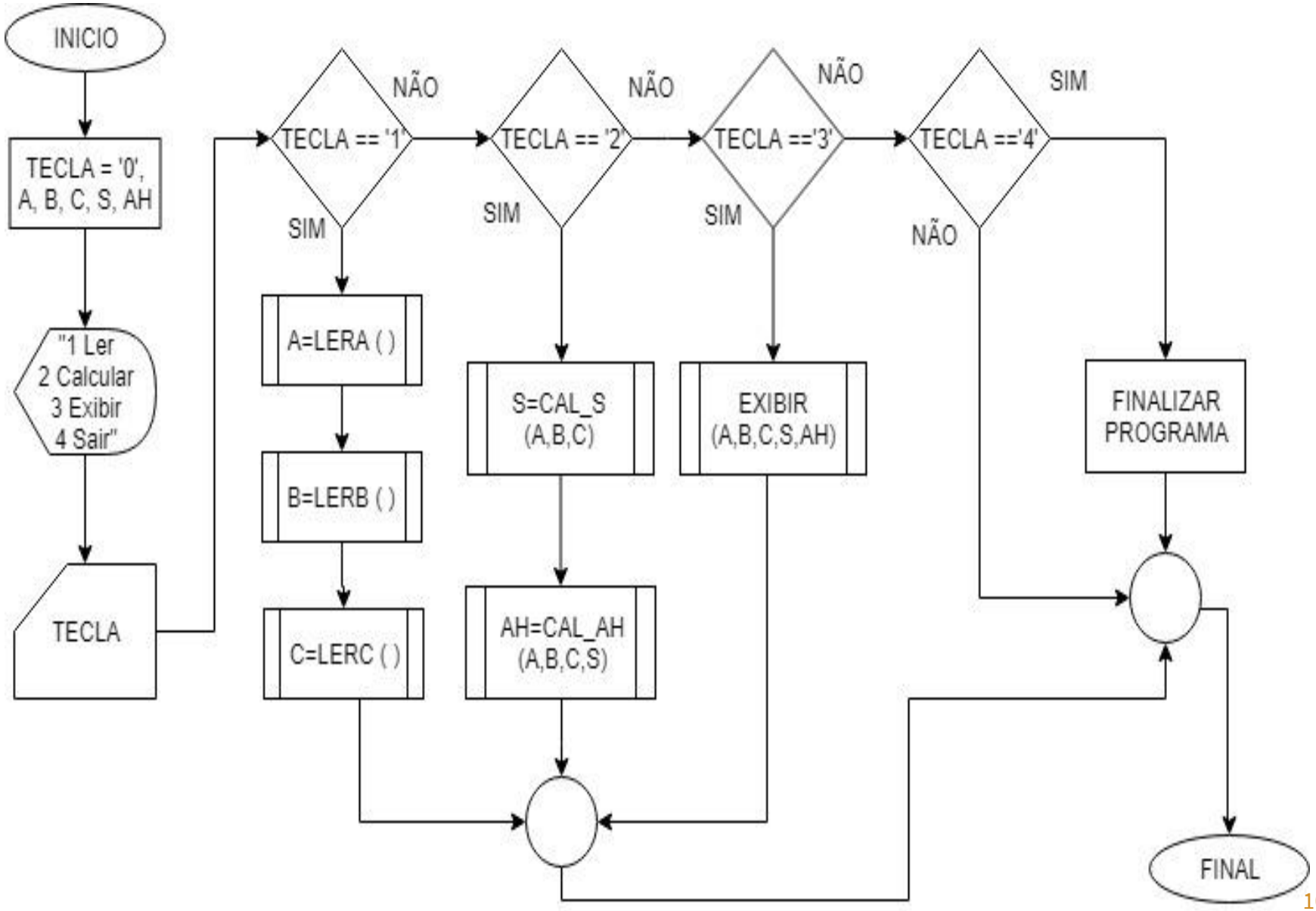
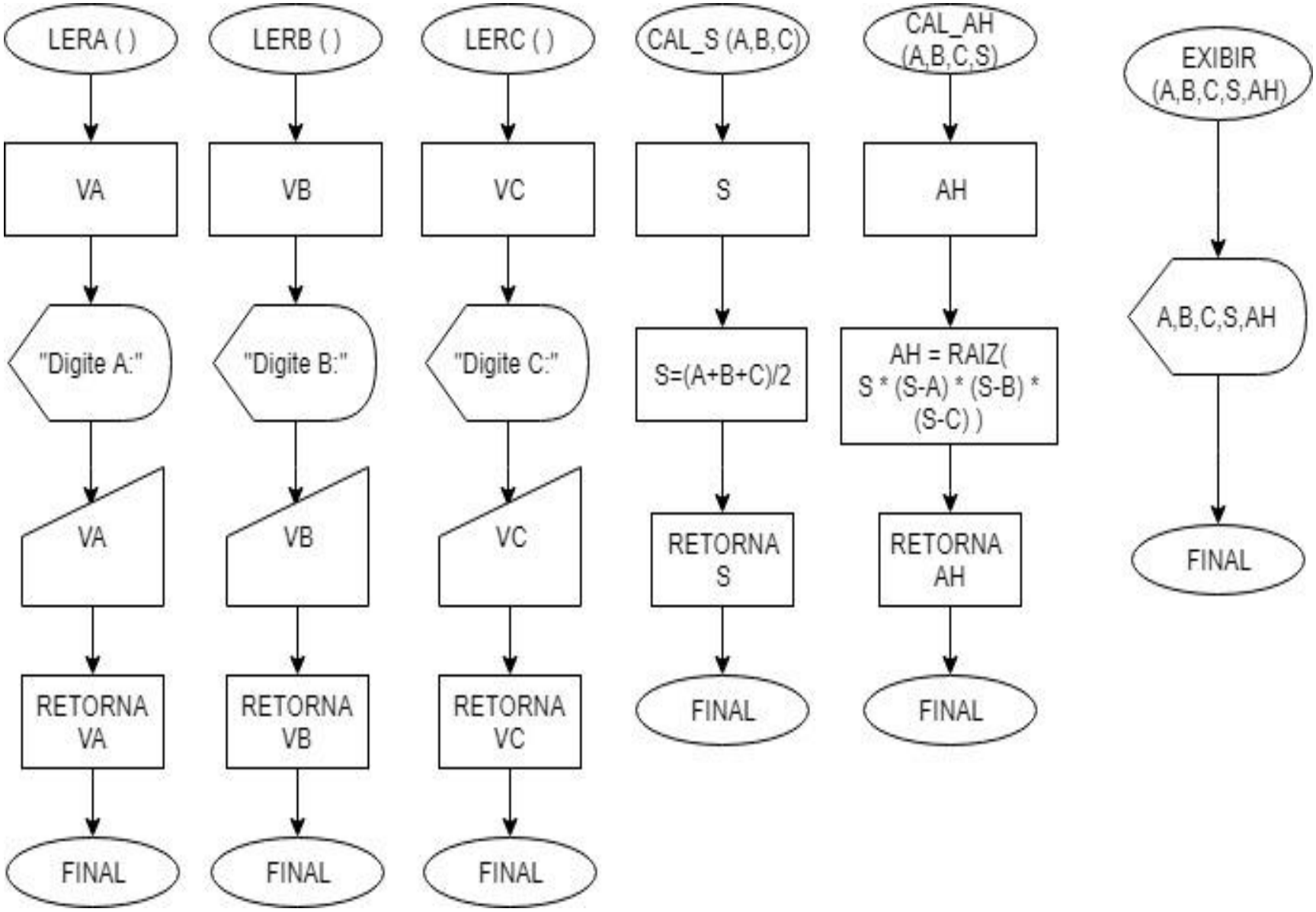


Diagrama nível 2 - sub rotinas



TAREFA DE FIXAÇÃO

a) Monte, compile e execute o programa de aprendizagem, comente todas as linhas e instruções do programa explicando cada subrotina no código para não esquecer, depois mexa no código do void **exibir ()** para inserir os comandos de formatação de saída da biblioteca iomanip.h: setfill(), setprecision(), fixed e setw(). Também coloque no início do codigo as declarações das sub-rotinas conforme foi passado pelo professor no vídeo da aula.

b) Faça o código fonte da atividade de **FIXAÇÃO**, a partir dos Diagramas níveis 1 e 2. Em vez de usar um switch case, use o comando IF ELSE para fazer o menu conforme o diagrama nível 1 main(), não esquecendo de formatar a tela de saída com os comandos da biblioteca iomanip.h. No código fonte faça a declaração das sub-rotinas no início do código e o código das subrotinas após a função main.

Observe que no diagrama não consta o laço de menu, entretanto, no código fonte, marque um ponto antes da exibição do menu chamado MENU e antes do return 0 que marca o final do código use o comando goto para executar a partir do ponto marcado como MENU.