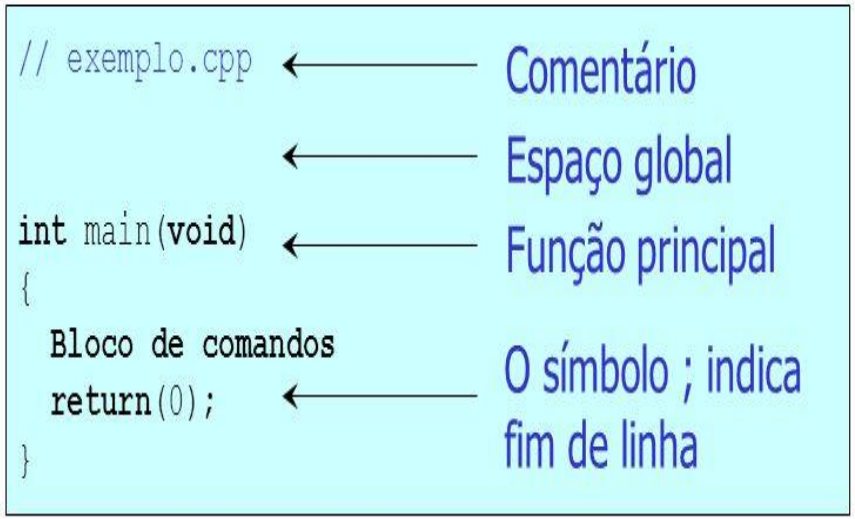
4
Variáveis Locais, Globais & Sub-Rotinas

Programação Estruturada

- Um programa C++ é dividido em funções:
  - As definições/declarações no interior funções são **locais**.
  - As definições/declarações fora das funções são **globais**.

- Todo programa precisa ter uma função chamada **main**
  - Esta função é a primeira a ser chamada quando um programa é executado.
  - Sintaxe:



Todo código fonte para ser executado deve ter a função **main ( )**, entretanto você poderá criar suas próprias sub rotinas que podem ser functions ou voids.

As sub rotinas podem ser criadas logo abaixo das diretivas do pré-processador e acima da função main ( ), exatamente no **espaço global** onde também serão criadas as **constantes** e **variáveis globais.**

Sub Rotina do Tipo VOID (Vazio)

Um programa é formado por sub-rotinas que podem ser VOIDS ou FUNCTIONS. **O VOID** é um código que ao ser executado, irá armazenar sempre VAZIO na memória global, ou seja, não armazena valores. O comando **return** em um **VOID** é utilizado para finalizar o código dentro da void.

PROGRAMA 1 - Primeira Parte:

```
#include "iostream"
#include "cstdlib"
using namespace std;


void exibir ( double n1,  double n2  )
{
   if ( n1==0 || n2==0 ) return;
     else
         cout<< "\n A média é "<< (n1 + n2) /2;


    cout<< endl;
    system("pause");
}
```

void exibir ( )

Os voids e functions podem ser criados na parte superior do código fonte logo após as diretivas do pré-processador e acima da função main ou ainda podem ser criados abaixo da função main, só que neste segundo caso as functions e voids deverão ser declaradas acima da função main.

Ao lado, no void **exibir** ( ), as variáveis n1 e n2, que foram declaradas entre parênteses são chamadas de argumentos e são variáveis de entrada do void.

Sub Rotina do Tipo VOID (Vazio)

PROGRAMA 1 - Parte Final:

```
int main()
{
exibir( 5  ,  9.3 );  // CHAMADA POR VALOR

double X = 4,  Y = 5.5 ;

exibir (X,Y );// CHAMADA POR REFERÊNCIA

system("pause");

return 0;
}
```

função int main ( )

Na função main(), **para executar o void exibir ( ), basta inserir o nome do void, no caso, exibir ( ),** enviando-se os argumentos necessários que são os dois valores, o 5 que será atribuído para n1 e o 9.3 que será atribuído para n2, é o que chamamos de **chamada por valor.**

Na segunda vez que o void exibir é chamado/executado, são enviadas as variáveis X e Y, o valor de X será atribuído a n1 e o valor de Y será atribuído a n2, é o que chamamos de **chamada por referência.**

Voce pode alterar o código ao lado pedindo para o usuário digitar os valores de X e Y pelo teclado e em seguida enviando essas variáveis para o void exibir.

Sub Rotina do tipo FUNCTION (não void)

Um código não void (function) armazenar valores, ou seja, serve para produzir e armazenar um único valor na memória global que poderá ser atribuído a uma variável ou impresso na tela. O comando **RETURN** será utilizado para armazenar o valor criado pela função, ou seja, o valor de retorno da função e também finalizar o código. Existem functions de **Leitura**, de **Cálculo** e de **Saída**. Para executar/chamar uma function deve-se criar uma variável e atribuir a function.

PROGRAMA 2:

```
#include "iostream"
#include "cstdlib"
using namespace std;

double lern1() {double n1;
cout << "Digite n1";
cin >> n1;  return n1;}

double lern2() {double n2;
cout << "Digite n2";
cin >> n2;  return n2;}

double calcMedia(double n1, double n2)
{ double media = (n1+n2)/2;
 return media;  }

int mostrar ( double media )   {
 cout << "\nMedia:" << media << endl;
 system("pause"); return 0;     }

int main()       {
double nota1, nota2, media;// variáveis para funções

int  imprimir; // variável para a função mostrar

 nota1 = lern1( ); // executa lern1()
 nota2 = lern2( ); // executa lern2()
 media = calcMedia(nota1, nota2); //executa calcMedia
 imprimir = mostrar(media); // executa void mostrar

return 0; } // fim do programa
```

O programa 3 possui duas funções de leitura, **lern1()** e **lern2()**, uma função de cálculo **calcMedia()** com dois argumentos **n1** e **n2**. A função **mostrar** recebe o argumento media é exibe o valor da média na tela.

2 REGRAS DE USO DE ARGUMENTOS

1 – Argumentos/Parâmetros, são variáveis locais de ENTRADA de um VOID ou FUNCTION, declaradas sequencialmente, uma a uma dentro dos parêntesis para receber valores de entrada, portanto não devem ser declaradas na área global, pois são locais.

2 – Na lista de Argumentos/Parâmetros de um VOID ou FUNCTION, não é necessário declarar **constantes**, **variáveis globais** e **variáveis de retorno** de fórmulas internas da FUNCTION ou VOID.

Recomendo: ***Assistam os vídeos Aula 3 Programação Estrutura em C++ partes 1 e 2***

Canal: Eliseu Lemes C++ (PlayList Aulas de C++)

Variáveis Locais, Globais & Constantes

Variáveis locais são as variáveis criadas dentro de um **Escopo Local**, isto é, dentro das sub rotinas e somente o código da própria sub rotina onde foi criada, poderá ter acesso aos valores dessas variáveis.

Variáveis Globais são as variáveis criadas na área do **Escopo Global,** exatamente onde deverão ser criadas também as **Constantes**, desta forma, as **Variáveis Globais e Constantes** compartilham seus valores dentro de todas as funções do programa, todos os códigos terão acesso livre a seus valores.

PROGRAMA 3 - Parte 1:

```
#include "iostream"
#include "cstdlib"
using namespace std;
double const PI =3.14;   // constante
double const G = 9.8;    // constante
double MEDIA1= 5; // variável global

void listar()  {
        cout<< "PI ="<< PI;
         cout<< "G ="<< G;
         cout<< "media1="<< MEDIA1;
          cout<< "media2="<< MEDIA2<< endl;//Erro
          system("pause");      }
```

void Listar( )

No escopo global foram criadas as constantes **PI, G** e a variável global **MEDIA1.**

O void **listar()** exibe corretamente as constantes PI e G, a variável global **MEDIA1** mas não consegue exibir a variável **MEDIA2**, pois somente o **int main** pode acessá-la, já que está variável foi criada dentro do int **main**.

Variáveis Locais, Globais & Constantes

PROGRAMA 3 - Parte Final :

```
int main( )
{
   double MEDIA2=9;
    /* A variável local MEDIA2
    não poderá ser usada dentro de outra
função ou void a não ser a função onde ela
foi criada localmente que neste caso é
função main(),  MEDIA2 é uma variável
local, esse código deverá ser alterado para
compilar e funcionar  */
      listar ( );
      return 0;
}
```

Como arrumar o programa 3:

Opção 1 - Crie um argumento no void listar: **listar (double MEDIA2)** e dentro do int **main** chame o void listar da seguinte forma: **listar (MEDIA2);**

Opção 2 - Simplemente comente o código: // double MEDIA2=9; em seguida declare a variável MEDIA2=9 como global.

Opção 3 - Declare a variável MEDIA2=9 localmente dentro do void listar ();

Atenção: Teste cada uma das opções acima para executar o código.

EXERCÍCIOS DE FIXAÇÃO

Fazer o código fonte juntamente com as declarações das funções dentro do programa (Quadro resumo de sub-rotinas): Exercícios I, J e L da página 26 do livro: Estudo Dirigido de Algoritmos. A criação de menu é opcional dentro do int main(). Execute as funções ou voids como foi feito nos **programas 1 e 2.**

NOTA:

1 - Crie funções para ler todas as variáveis de leitura, crie funções para fazer o cálculo de todas as fórmulas e crie um void para mostrar os resultados.

2 - O Quadro Resumo de Sub-rotinas deverá ser feito dentro do próprio código conforme o explicado pelo professor na transmissão da aula.