5
Operações com SubRotinas

Estudando o tipo void

Programação estruturada é quando um programa contém sub-rotinas que podem ser functions ou voids. **O tipo void** é um pedaço de código/subrotina que têm o objetivo de executar uma lista de instruções qualquer, ao final das execuções, o valor que irá restar na **memória global** será sempre void, ou seja, vazio.

Programa 1

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

void exibir (string nomedigitado )
{
 cout << "\nOlá " << nomedigitado;
 cout << "\nbem vindo à aula on-line!\n";
}

int main ()
{
 string nome;
 setlocale(LC_ALL, "Portuguese");

 cout << "\nDigite o Nome:";
 getline(cin, nome);

 exibir ( nome );

return 0; }
```

Logo após a diretiva using é criado o void **exibir (nomedigitado),** com o argumento **nomedigitado** do tipo string que será a variável de entrada do void.

Na função **main()**, após o usuário digitar o nome de uma pessoa, a função **getline (cin, nome)** que depende da biblioteca "string" lê o nome digitado e armazena na variável do tipo string **nome**.

Na próxima linha, o void **exibir ( )** é chamado enviando-se a variável **nome** para dentro do void, esta variável **nome** se tornara no argumento **nomedigitado,** que será utilizado para mostrar o nome digitado na tela.

Comportamento da Memória Programa 1

Tudo que é criado dentro de um código é armazenado na memória, **as variáveis globais e locais, os argumentos, os nomes dos voids e os nomes das funções**, etc. Por exemplo, no programa 1, o void **exibir (nomedigitado),** a variável **nome**, que é a variável local de leitura para ler o nome que será digitado e variável local, **nomedigitado**, que é o argumento de entrada do void exibir().

Supondo-se que o usuário digite o nome da pessoa **"José da Silva"** no **programa1**, ao final da execução podemos identificar quais os últimos valores que ocuparam espaço de memória de acordo com o quadro abaixo:

Identificadores gravados na memória após a execução do Programa1				
Nome_identificador	TIPO	ULTIMO VALOR	DESCRIÇÃO	ESCOPO
main()	int	0	função	Global
nome	string	"Jose da Silva"	variável	local
exibir(nomedigitado)	void	vazio	procedimento	Global
nomedigitado	string	"Jose da Silva"	variável	local

Estudando o tipo não void (function)

A function (**tipo não void)** é um pedaço de código que têm o objetivo de armazenar um valor ao final de seu processamento. Esse valor armazenado será de acesso global como uma variável global, pois o nome da função será uma variável global temporária que só ficará na memória durante o seu tempo de execução.

Programa 2

```
#include "iostream"
#include "iomanip"
using namespace std;
double const taxaaumento = 0.1;

double novosalario (double salarioatual)
{ double aumento;
 aumento = (salarioatual * taxaaumento) +
salarioatual;
 return aumento;
}

int main () {
 setlocale(LC ALL,"Portuguese");
 double salatual, novosal;

cout << "\nSalário atual:";
cin >> salatual;

novosal = novosalario (salatual);

cout << setfill('.');
cout << fixed << setprecision(2);

cout << "\nSalário Novo:";
cout << setw(11) << novosal << endl;
return 0; }
```

Logo após a diretiva using, é criada a função **double novosalario(salarioatual),** para calcular o acréscimo de 10% a um salário digitado qualquer.

Na função **main()**, são criadas as variáveis locais **salatual** para ler o salario atual e **novosal** para armazenar o valor calculado pela função **novosalario().**

Após o usuário digitar o valor do salário de alguém, a função **novosalario()** recebe o argumento **salatual** e é chamada pela variável **novosal** que terá o valor do aumento calculado de 10% sobre o salário atual que será impresso na tela.

Comportamento da Memória Programa2

Ao final da execução do programa, as funções terão armazenados valores resultantes de seus códigos internos que foram armazenados pelo comando **return** de cada função. Supondo-se que o usuário digite o salário de 100 reais, vejamos como ficará a memória ao final da execução do **programa2** de acordo com a tabela abaixo:

Identificadores gravados na memória após a execução do Programa2				
Nome_identificador	TIPO	ULTIMO VALOR	DESCRIÇÃO	ESCOPO
taxaaumento	double	0.1	constante	global
novosalario(salarioatual)	double	110	function	global
salarioatual	double	100	variável	local
aumento	double	110	variável	local
main()	int	0	function	global
salatual	double	100	variável	local
novosal	double	110	variável	local

Considerações sobre void e function

1) a) O que é um código void ? É um código formado por um conjunto de instruções que ao serem executadas deixarão um espaço vazio na memória global, portanto não têm o objetivo de armazenar valores. O comando **return** em um void servirá apenas para finalizar o código. **b)Como se executa um void ?** Basta digitar o nome dele como se fosse um programa.

Exemplo chamada de um void:

```
void exibir() {  cout << "Olá mundo"; }
void ler (int x) { y=x; }  int main(){     exibir(  );    ler(20);   int z =5;    ler(z); }
```

2) a) O que é uma função? É um conjunto de instruções que ao serem executadas irão produzir e armazenar apenas 1 (um) único valor na memória global. O comando **return** em uma função será utilizado para fazer o armazenamento do valor final da função (retorno). b) Como executar uma função? Deve-se **criar uma variável, de preferência local** e atribuir à função.

Exemplo chamada de uma função:

```
double valormedia( double n1, double n2) {   return (n1 + n2)/2;   }

int main () {    double minhamedia = valormedia (10, 5);

                 double x=10, y=5;        minhamedia = valormedia(x,y); }
```

TAREFA DE FIXAÇÃO

Faça o CÓDIGO FONTE de um programa contendo menu de controle, opções 1 ler valores, 2 calcular média, 3 exibir tudo e 4 sair. Deverá conter 3 funções para ler três valores, uma função para calcular a média geométrica dos três valores e um void mostrar para mostrar os três valores e média geométrica na tela. No código faça as declarações das sub-rotinas no início do código fonte. Não se esqueça de utilizar as funções da biblioteca iomanip setfill(), setprecision e setw().

MEDIA GEOMÉTRICA = RAÍZ CÚBICA (V1 x V2 x V3)