Conceitos Básicos da Linguagem C++

Sumário

- As Origens da Linguagem C++;
- Primeiro Programa em C++;
- Declaração e Leitura de Variáveis;
- Estruturas de Seleção;
  - if;
  - switch;
- Estruturas de Repetição;
  - while;
  - do/while;
  - for;

Sumário

- Os comandos `break e continue`;
- Funções:
  - Protótipo de Funções;
  - Arquivos de Cabeçalho;
  - Recursão;
  - Função *inline*;
  - Passagem de Parâmetros por Referência;
  - Sobrecarga de Funções;
  - Gabaritos de Funções;
- Array;
- Bibliografia.

As Origens da Linguagem C++

- C++ é uma evolução de C, que evoluiu de duas linguagens anteriores, BCPL e B.
- A linguagem BCPL influenciou uma linguagem chamada B, inventada por Ken Thompson;
- Na década de 70, B levou ao desenvolvimento da **linguagem C**.
- A linguagem de programação C foi originalmente projetada para ser implementada no sistema operacional UNIX em um DEC PDP-11;

As Origens da Linguagem C++

- O padrão C foi a versão fornecida com o sistema operacional UNIX versão 5;
- Com a popularidade dos microcomputadores, um grande número de implementações de C foi criada;
- Para remediar a falta de padrão da linguagem C, o ANSI (American National Standards Institute) estabeleceu, em 1983, um comitê para criar um padrão definitivo da linguagem C

As Origens da Linguagem C++

- C++, uma extensão de C, foi desenvolvida por Bjarne Stroustrup no início dos anos 80 no Bell Laboratories;
- C++ é uma linguagem híbrida - é possível programar tanto no estilo C como no estilo orientado a objetos, ou em ambos;
- A linguagem C++ trouxe várias características que melhoram a linguagem C, sobretudo, os recursos para suportar a Programação Orientada a Objetos;

Primeiro Programa em C++

- Analisemos, agora, um simples programa em C++:

primeiroCPP.cpp

```
//============================================================================
// Name        : primeiroCPP.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
    return 0;
}
```

Problems | Tasks | Console | Properties

```
<terminated> primeiroCPP.exe [C/C++ Local Application] C:\Documents and Settings\Leonardo\workspace\primeiroCPP\Debug\primeiroCPP.exe (15/08/07 10:04)
!!!Hello World!!!
```

Primeiro Programa em C++

Nome do programa
(C plus plus daí *.cpp)

Comentário de linha
única. Utilizado para
documentar o código

primeiroCpp.cpp

```
//============================================================================
// Name        : primeiroCpp.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
    return 0;
}
```

Usa o ambiente de nomes std

Diretiva para inclusão
do arquivo cabeçalho
<iostream> no programa

Primeiro Programa em C++

Função principal do programa

primeiroCpp.cpp

```
//============================================================================
// Name        : primeiroCpp.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
    return 0;
}
```

Operador que insere o valor a direita do operador na saída

Objeto que corresponde ao *stream* padrão de saída

Manipulador de *stream* que gera uma nova linha e esvazia o buffer de saída

Primeiro Programa em C++

- Considerações:
  - Comentário contendo muitas linhas começa com /* e termina com */
  - As linhas iniciadas com # são processadas pelo pré-processador antes do programa ser compilado;
  - O ambiente de nomes define um escopo onde identificadores e variáveis são colocados;
  - A E/S em C++ ocorre em *streams* de bytes. Um *stream* é simplesmente uma seqüência de bytes. Por exemplo, em operações de entrada onde os bytes fluem de um dispositivo (teclado, HD, conexão de rede) para a memória principal.

Primeiro Programa em C++

- Na linguagem C o envio de dados para a saída padrão do computador era feita pela função printf;

  printf(string_de_controle, lista_de_argumentos)

- A função printf recebe uma string e, possivelmente, uma lista de argumentos para tratar;

- Na linguagem C++ a *stream* a ser apresentada na saída padrão é tratada pelo objeto cout;

Primeiro Programa em C++

- O manipulador de *stream* endl pode ser substituído pela sequência de scape \n, mas deve vir entre aspas duplas. Veja outras seqüências de scape comuns:

Código	Significado
\n	Nova linha
\r	Retorno de carro
\t	Tabulação horizontal
\"	Aspas duplas
\'	Aspas simples
\\	Barra invertida
\v	Tabulação vertical
\a	Alerta

Declaração e Leitura de Variáveis

Objeto cin de entrada de *stream* (do ambiente de nomes std)

Operador >> de extração de stream

aula02Calculadora.cpp

```
#include <iostream>
using namespace std;

int main() {
    int num1, num2, mult;
    cout << "*************************************************" << endl;
    cout << "* Este programa soma, subtrai e multiplica dois números *" << endl;
    cout << "*************************************************" << endl;
    cout << "Digite o primeiro número: ";
    cin >> num1;
    cout << "Digite o segundo número: ";
    cin >> num2;
    cout << "A soma de " << num1 <<" com " << num2 <<" é: " << num1+num2 << endl;
    cout << num1 <<" - " << num2 <<" é: " << num1-num2 << endl;;
    mult = num1 * num2;
    cout << num1 <<" vezes " << num2 <<" é: " << mult << endl;;
    return 0;
}
```

Declaração de variáveis

Declaração e Leitura de Variáveis

- Os tipos de dados primitivos da linguagem C++ são:

Tipo de Dados	Considerações
long double	
double	
float	
unsigned long int	Sinônimo de unisigned long
long int	Sinônimo de long
unsigned int	Sinônimo de unsigned
int	
unsigned short int	Sinônimo de unsigned short
short int	Sinônimo de short
unsigned char	
char	
bool	false torna-se 0, true torna-se 1

Declaração e Leitura de Variáveis

- Precedência dos Operadores Aritméticos:

Operadores	Ordem de Avaliação
( )	Prioridade Alta. Parêntese mais internos e "no mesmo nível" da esquerda para a direita
*, / ou %	Prioridade Média. Se houver vários, da esquerda para a direita.
+ ou -	Prioridade Baixa. Se houver vários, da esquerda para a direita.

Estrutura de Seleção – if

aula02Calculadora.cpp

```
#include <iostream>
using namespace std;

int main() {
    int num1, num2;
    cout << "*************************************************************" << endl;
    cout << "* Este programa informa se um número é divisel por outro *" << endl;
    cout << "*************************************************************" << endl;
    cout << "Digite o primeiro número: ";
    cin >> num1;
    cout << "Digite o segundo número: ";
    cin >> num2;
    if( num1%num2 == 0)
        cout << num1 <<" é divisível por " << num2 << endl;
    else
        cout << num1 <<" não é divisível por " << num2 << endl;;
    return 0;
}
```

Estrutura de Seleção - switch

Funciona como ou ('A' ou 'a')

Aula02.cpp

```
#include <iostream>
using namespace std;

int main() {
    char op;
    cout << "***************" << endl;
    cout << "* A * Arquivo *" << endl;
    cout << "* E * Editar  *" << endl;
    cout << "* S * Sair    *" << endl;
    cout << "***************" << endl;
    cin >> op;
    switch(op)
    {
        case 'A': case 'a': cout << "**** Menu Arquivo ****"; break;
        case 'E': case 'e': cout << "**** Menu Editar ****"; break;
        case 'S': case 's': cout << "O Programa está sendo finalizado..."; break;
        default: cout << "Opção Inválida... O programa foi finalizado"; break;
    }
    return 0;
}
```

Estrutura de Repetição – while

++letra (pré-incremento)

Saída: b c d e f g h i . . .

Aula02.cpp

```
#include <iostream>
using namespace std;

int main() {
    char letra='a';
    cout << "* * * * * * * * * * * * * * * * * * * * * * * * * *" << endl;
    cout << "* Este programa imprime as letras do alfabeto *" << endl;
    cout << "* * * * * * * * * * * * * * * * * * * * * * * * * *" << endl << endl;
    while(letra <= 'z')
        cout<< letra++ << " ";
    return 0;
}
```

Problems | Tasks | Properties | Console

<terminated> HelloWorld.exe [C/C++ Local Application] C:\Documents and Settings\Leonardo\workspace\Hello World\Debug\HelloWorld.exe

```
* * * * * * * * * * * * * * * * * * * * * * * * * *
* Este programa imprime as letras do alfabeto *
* * * * * * * * * * * * * * * * * * * * * * * * * *

a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Estrutura de Repetição – do while

Aula02.cpp

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

int main() {
    float a, b, c, delta;
    cout << "*************************************************************************" << endl;
    cout << "* Este prog calcula as raizes de uma equação do 2° grau f(x)=a.x.x + b.x + c *" << endl;
    cout << "*************************************************************************" << endl << endl;
    do{
        if(a==0)
            cout << "A equação f(x)= " << b << ".x + " << c << " não é do 2° grau. Tente Novamente!" << endl;
        cout << "Informe o valor de a: "; cin >> a;
        cout << "Informe o valor de b: "; cin >> b;
        cout << "Informe o valor de c: "; cin >> c;
    }while(a==0);
    delta = ((pow(b,2))-(4*a*c));
    if(delta <0)
        cout << "A equação f(x)= "<< a <<".x.x +("<< b <<").x + ("<< c <<") não possue raiz real";
    else
        if(delta ==0)
            cout << "A raiz da equação é: "<< -b/2*a << endl;
        else
            cout << "As raizes da equação são: x'= " << setprecision(5) << -b+sqrt(delta)/2*a
                 <<" e x''= " << setprecision(5) << -b-sqrt(delta)/2*a;
    return 0;
}
```

Biblioteca Matemática

Biblioteca de manipulação de I/O

Estrutura de Repetição – for

Aula02.cpp

```
#include <iostream>
using namespace std;

int main() {
    bool op=0;
    char teste; float a=0, b=0;
    cout << "*****************************************************************" << endl;
    cout << "* Este prog informa se o débito da cantina atingiu R$ 100,00 *" << endl;
    cout << "*****************************************************************" << endl << endl;
    for(float conta=a, compra=b;( (conta<100) && op==0); a=conta, b=compra)
    {
        cout << "Informe o valor da compra: ";
        cin >> compra;
        conta+=compra;
        cout << "Deseja realizar uma nova compra? (S/N): ";
        cin >> teste;
        if (teste == 'N' || teste== 'n')
            op = 1;
    }
    if(a<100)
        cout << "O débito está em R$ " << a << endl;
    else
    {
        a -=b;
        cout << "A compra não pode ser efetuada... limite de R$ 100,00 atingido" << endl
             << "Seu débito continua em R$ "<< a;
    }
    return 0;
}
```

Tipo Bool (Só aceita 1 ou 0, Verdadeiro ou Falso)

Conta e compra são variáveis locais

a e b são variáveis globais

Operadores Lógicos && (AND)

a = a - b;

É permitida a quebra de linha

Estruturas de Seleção

- Precedência e Associatividade de Operadores:

Operadores	Associatividade	Tipo
( )	esquerda para a direita	Parênteses
++ -- static_cast<type>()	esquerda para a direita	Unário (pós-fixado)
++ -- + -	direita para a esquerda	Unário (pré-fixado)
* / %	esquerda para a direita	Multiplicativo
+ -	esquerda para a direita	Aditivo
<< >>	esquerda para a direita	Inserção/extração
< <= > >=	esquerda para a direita	Relacional
== !=	esquerda para a direita	Igualdade
&&	esquerda para a direita	E lógico
||	esquerda para a direita	OU lógico
?:	direita para a esquerda	Condicional
= += -= /= %=	direita para a esquerda	Atribuição
,	esquerda para a direita	Vírgula

Os comandos break e continue

- O comando break provoca uma saída imediata da estrutura (for, while,, if, etc);

- O comando continue, quando executado em uma estrutura (while, for ou do/while), salta os comandos restantes no corpo dessa estruturae prossegue com a próxima repetição do laço, vejamos:

```
for(int cont=0; cont<=10; cont++)
{
    if(cont == 3)
        continue;
    cout << cont << " ";
}
```

Saída

0 1 2 4 5 6 7 8 9 10

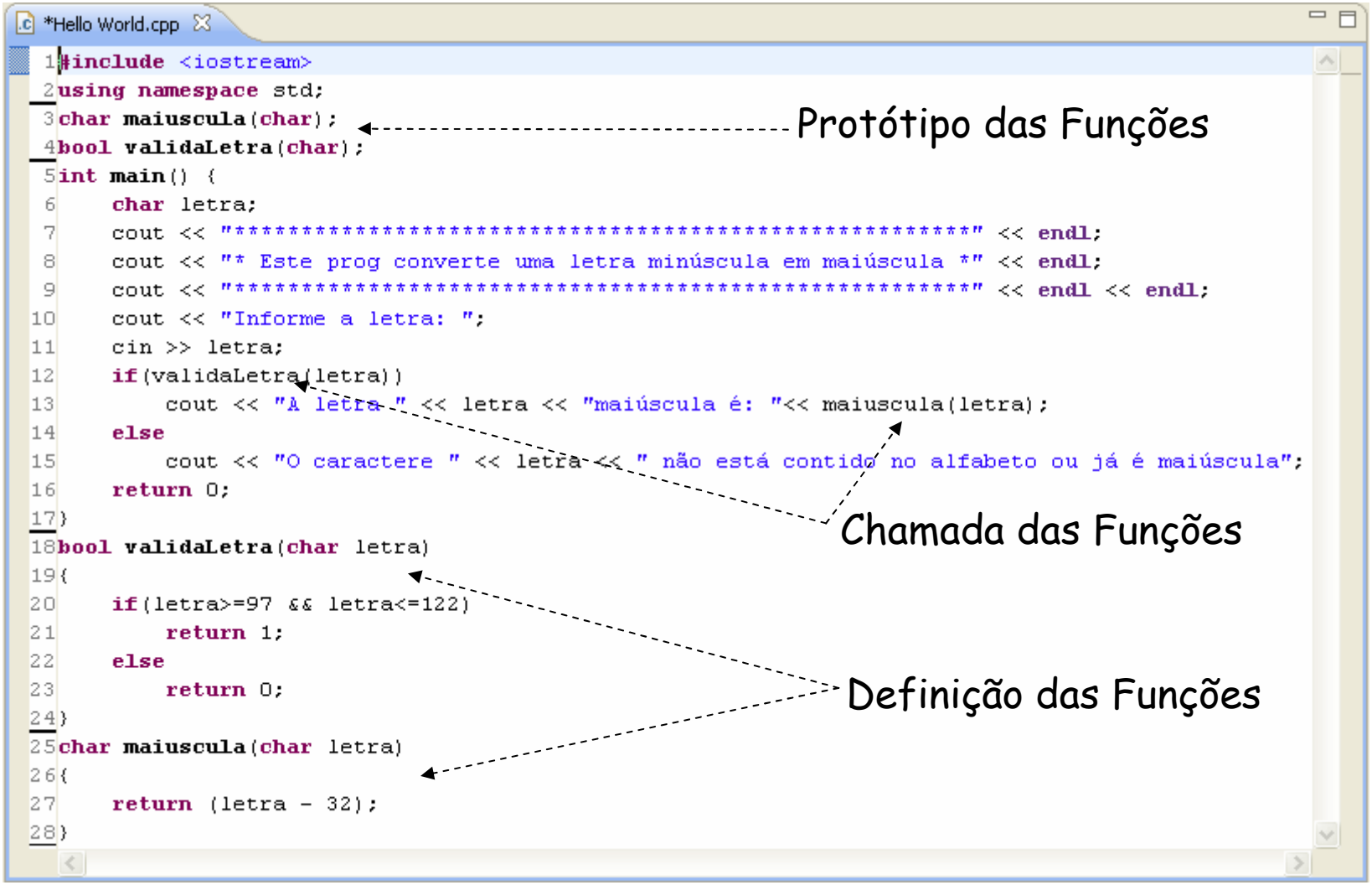
Funções

- A experiência tem mostrado que a melhor maneira de desenvolver e manter um programa grande é contituí-lo a partir de pequenas partes ou componentes;

- As funções vêm para possibilitar ao programador a modularização do programa facilitando a implementação, a operação e as manutenção;

- Os programas em C++ são escritos tipicamente combinando-se funções novas que o programador escreve com "funções pré-empacotadas" disponíveis na biblioteca padrão de C++;

Protótipo de Funções



Arquivos de Cabeçalho

- Os arquivos de cabeçalho que terminam em `.h` são arquivos de cabeçalho no "estilo antigo", que foram substituídos pelos arquivos de cabeçalho da biblioteca padrão C++;

- Para os arquivos de cabeçalho definidos pelo programador, os mesmo devem terminar com `.h` ;

- Os arquivos devem ser incluídos pelo programador através da diretiva de pré-processador `#include`;

Arquivos de Cabeçalho

Arquivos de Cabeçalho da Biblioteca Padrão	Considerações
<cctype>	Contém protótipos para funções examinarem caracteres em busca de determinadas propriedades dos caracteres. Substitui <ctype.h>
<cmath>	Contém protótipos de funções da biblioteca matemática. Substitui <math.h>
<cstdio>	Contém protótipos de funções da biblioteca padrão de Entrada/Saída e as informações utilizadas por elas. Substitui <stdio.h>
<cstring>	Contém protótipos de funções para processamento de strings. Substitui <string.h>
<iostream>	Contém protótipos de funções para as funções padrões de Entrada/Saída. Substitui <iostream.h>
<iomanip>	Contém protótipos de funções para os manipuladores de streams que permitem formatação de streams de dados. Substitui <iomanip.h>
<fstream>	Contém protótipos de funções para funções que executam operações de entrada a partir de arquivos de disco e saída para arquivos em disco. Substitui <fstream.h>

Recursão

Hello World.cpp

```
1#include <iostream>
2using namespace std;
3long int fibonacci(int);
4int main() {
5    int termo;
6//  long int resultado;
7    cout << "*****************************************************" << endl;
8    cout << "* Este prog calcula Fibonacci usando Recursividade *" << endl;
9    cout << "*****************************************************" << endl << endl;
10   cout << "Informe o termo da série: ";
11   cin >> termo;
12   cout << "Fibonacci (" << termo << ") = " << fibonacci(termo);
13   return 0;
14}
15long int fibonacci(int x)
16{
17   if ( (x == 0) || (x == 1) )
18       return x;
19   else
20       return (fibonacci(x-1) + fibonacci(x-2));
21}
```

Saída no Console

```
*****************************************************
* Este prog calcula Fibonacci usando Recursividade *
*****************************************************

Informe o termo da série: 10
Fibonacci (10) = 55
```

Funções *inline*

- As funções inline tem a finalidade de diminuir o overhead nas chamadas de função, principalmente para funções pequenas;

Hello World.cpp

```
1#include <iostream>
2using namespace std;
3inline long int areaQuadrado(int x) {return x*x;}
4int main() {
5    int lado;
6    cout << "*****************************************" << endl;
7    cout << "* Este prog calcula a área de um quadrado *" << endl;
8    cout << "*****************************************" << endl << endl;
9    cout << "Informe o lado do quadrado: ";
10   cin >> lado;
11   cout << "A área do quadrado de lado " << lado << " é: " << areaQuadrado(lado);
12   return 0;
13}
14
```

O qualificador inline "aconselha" o compilador a gerar uma cópia do código da função no próprio local.

Passagem de Parâmetros por Referência

Hello World.cpp

```
1 #include <iostream>
2 using namespace std;
3 inline void funcao01(char ch)  {ch++;};
4 inline void funcao02(char &ch) {ch++;};
5 int main() {
6     char letra;
7     cout << "************************************************" << endl;
8     cout << "* Este prog mostra a próxima letra do alfabeto *" << endl;
9     cout << "************************************************" << endl << endl;
10    cout << "Informe a letra: ";
11    cin >> letra;
12    funcao01(letra);
13    if((letra=='Z')||(letra=='z'))
14        cout << "A próxima letra é: " << letra;
15    else
16        cout << "A próxima letra é: " << letra;
17    return 0;
18 }
19
```

Saída no Console

```
************************************************
* Este prog mostra a próxima letra do alfabeto *
************************************************

Informe a letra: z
A próxima letra é: z
```

Sobrecarga de Funções

- C++ possibilita que sejam definidas várias funções como o mesmo nome, desde que estas funções tenham conjuntos de parâmetros diferentes;

Hello World.cpp

```
1 #include <iostream>
2 using namespace std;
3 int areaQuadrado(int x) {return x*x;}
4 float areaQuadrado (double y) {return y*y;}
5 int main() {
6     cout << "*************************************************************************" << endl;
7     cout << "* Este prog calcula a área de um quadrado de lado 10 e outro de lado 10.5 *" << endl;
8     cout << "*************************************************************************" << endl << endl;
9     cout << "A área do quadrado de lado 10 é: " << areaQuadrado(10) << endl;
10    cout << "A área do quadrado de lado 10.5 é: " << areaQuadrado(10.5) << endl;
11    return 0;
12 }
```

```
*************************************************************************
* Este prog calcula a área de um quadrado de lado 10 e outro de lado 10.5 *
*************************************************************************

A área do quadrado de lado 10 é: 100
A área do quadrado de lado 10.5 é: 110.25
```

Gabarito de Funções

- Caso a lógica das funções sobrecarregadas mude apenas quanto ao tipo dos dados, é mais conveniente usar os gabaritos de funções;

- Com gabaritos o C++ gera automaticamente a função específica para tratar do tipo da chamada da função;

- Os gabaritos começam com a palavra-chave `template` seguida por uma lista de parâmetros de tipo formais;

Gabarito de Funções

Hello World.cpp

```
1#include <iostream>
2using namespace std;
3
4template < class T >
5T proximo (T valor)
6{
7    T prox = ++valor;
8    return prox;
9}
10int main() {
11    char ch;
12    int i;
13    float f;
14    double d;
15    cout << "************************************************************" << endl;
16    cout << "* Este prog demostra o comportamento dos gabaritos em C++ *" << endl;
17    cout << "************************************************************" << endl << endl;
18    cout << "informe um caracter: "; cin >> ch;
19    cout << "informe um inteiro: "; cin >> i;
20    cout << "informe um 'float': "; cin >> f;
21    cout << "informe um 'double': "; cin >> d;
22    cout << "O proximo caracter é: " << proximo(ch) << endl
23         << "O proximo inteiro é: " << proximo(i) << endl
24         << "O proximo float é: " << proximo(f) << endl
25         << "O proximo double é: " << proximo(d) << endl;
26    return 0;
27}
```

Este gabarito de função declara um parâmetro de tipo formal único, T.

Quando o compilador faz uma chamada à função proximo, o tipo dos dados passados para proximo substitui T ao longo de toda a definição do gabarito.

Gabarito de Funções

- Saída no console do programa anterior:

```
************************************************************
* Este prog demostra o comportamento dos gabaritos em C++ *
************************************************************

informe um caracter: a
informe um inteiro: 1
informe um 'float': 1.5
informe um 'double': 2.9
O proximo caracter é: b
O proximo inteiro é: 2
O proximo float é: 2.5
O proximo double é: 3.9
```

Array

- Os Arrays são grupos de posições de memória consecutivas, todas de mesmo nome e mesmo tipo;

Hello World.cpp

```
1#include <iostream>
2#include <iomanip>
3using namespace std;
4
5int main() {
6    const int tam=10;
7    int valores[ tam ] = {5, 9, 3, 16, 7, 10, 15, 1, 8, 4};
8    cout << "****************************************" << endl;
9    cout << "* Este prog introduz o conceito de Array *" << endl;
10   cout << "****************************************" << endl << endl;
11   cout << "Elemento " << setw(13) << "Valor"
12        << setw(17) << "Histograma" << endl;
13   for(int i=0; i< tam; i++)
14   {
15       cout << setw(7) << i << setw(13) << valores[i] << setw(9);
16       for(int j=0; j<valores[i]; j++)
17           cout << '*';
18       cout << endl;
19   }
20   return 0;
21}
```

Array

- Saída no Console para o programa anterior:

```
****************************************
* Este prog introduz o conceito de Array *
****************************************

Elemento          Valor        Histograma
      0             5          *****
      1             9          *********
      2             3          ***
      3            16          ****************
      4             7          *******
      5            10          **********
      6            15          ***************
      7             1          *
      8             8          ********
      9             4          ****
```

Bibliografia

- SANTOS, R. *Introdução à programação orientada a objetos usando Java,* Editora Campus. 1ª ed. Rio de Janeiro: 2003.

- Deitel, H. M. & Deitel, P. J. *C++: como programar,* Editora Bookman. 3ª ed. Porto Alegre: 2001.