GABARITOS DE FUNÇÕES (TEMPLATES)

Templates

- Permite criar funções genéricas (podendo ser considerados Generalizadores de Tipos) que admitem qualquer tipo de dado como parâmetro e retorna um valor sem ter que sobrecarregar a função com todos os tipos de dados possíveis.
- Isso é muito útil para definir funções que podem receber **parâmetros de tipos diferentes, mas que possuam lógicas de programação idênticas**.
- Os parâmetros de tipo formais são tipos primitivos ou tipos definidos pelo programador, usados para especificar os tipos dos parâmetros da função.

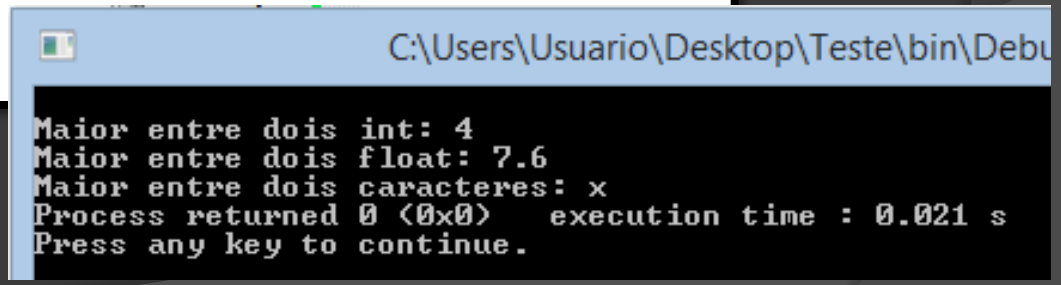
Templates

```
1   #include <iostream>
2   using namespace std;
3   #include <locale.h>
4
5   template <typename Tipo>
6   Tipo soma (Tipo a, Tipo b) {
7       return (a + b);
8   }
9
10  int main() {
11      setlocale(LC_ALL, "Portuguese");
12
13      cout<<endl<<"Soma de dois int: "<<soma(2 , 4);
14      cout<<endl<<"Soma de dois float: "<<soma(2.3 , 7.6);
15      cout<<endl<<"Soma de dois caracteres: "<<soma('1' , 'x');
16
17      return 0;
18  }
```

```
C:\Users\Usuario\Desktop\Teste\bin\Debug\Teste.exe

Soma de dois int: 6
Soma de dois float: 9.9
Soma de dois caracteres: ©
Process returned 0 (0x0)   execution time : 0.019 s
Press any key to continue.
```

Templates

```
1   #include <iostream>
2   using namespace std;
3   #include <locale.h>
4
5   template <typename Tipo>
6   Tipo maior (Tipo a, Tipo b) {
7       return (a>b? a : b);
8   }
9
10  int main() {
11      setlocale(LC_ALL, "Portuguese");
12
13      cout<<endl<<"Maior entre dois int: "<<maior(2 , 4);
14      cout<<endl<<"Maior entre dois float: "<<maior(2.3 , 7.6);
15      cout<<endl<<"Maior entre dois caracteres: "<<maior('x' , 'c');
16
17      return 0;
18  }
```



Templates

```
1  #include <iostream>
2  using namespace std;
3  #include <locale.h>
4  #define TAM 5
5
6  template <typename X>
7  void leituraVetor (X vetor[TAM]) {
8      int i;
9      for (i=0; i<TAM; i++) {
10         cout<<"Vetor na posição "<<i<<": ";
11         cin>>vetor[i];
12     }
13 }
14
15 template <typename X>
16 void escritaVetor (X vetor[TAM]) {
17     int i;
18     cout<<endl<<endl;
19     for (i=0; i<TAM; i++) {
20         cout<<vetor[i]<<"\t";
21     }
22 }
```

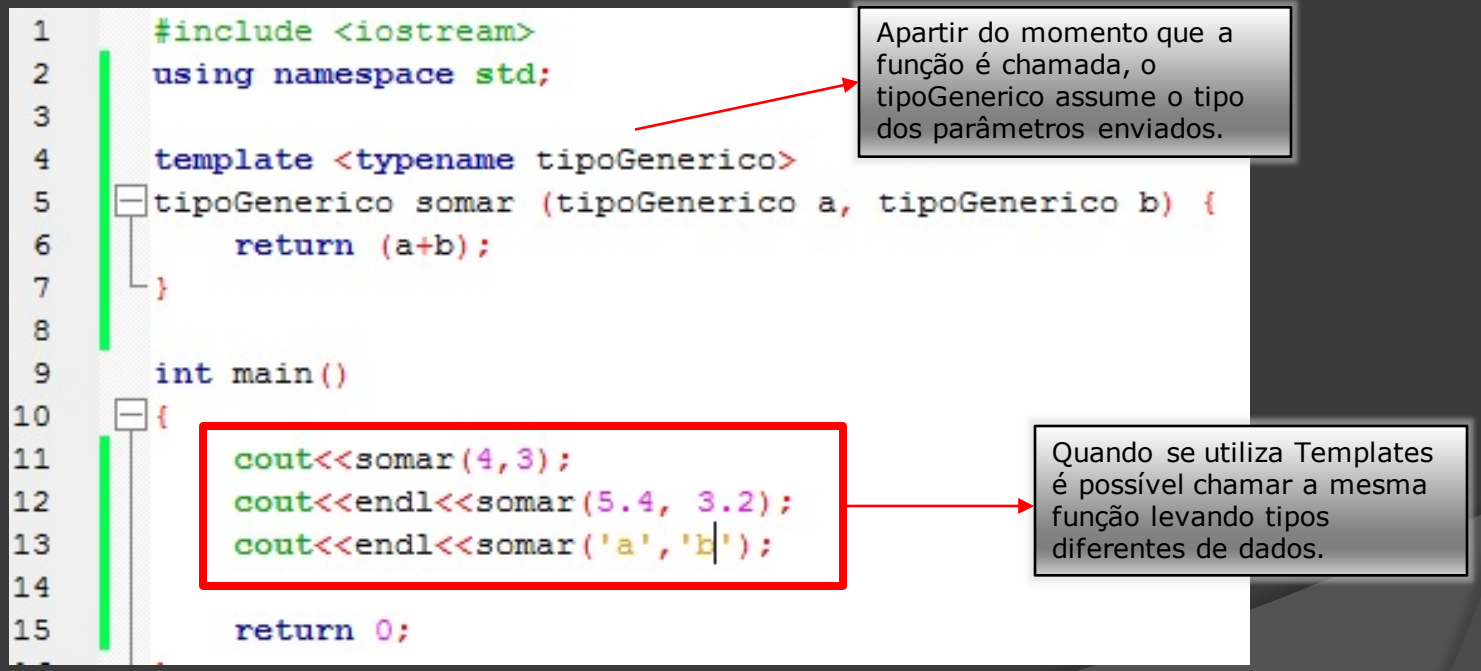
```
23
24 int main() {
25     setlocale(LC_ALL, "Portuguese");
26
27     int vetInt[TAM];
28     leituraVetor(vetInt);
29     escritaVetor(vetInt);
30
31     cout<<endl<<endl;
32     string vetChar[TAM];
33     leituraVetor(vetChar);
34     escritaVetor(vetChar);
35     return 0;
36 }
```

```
C:\Users\Usuario\Desktop\Teste\bin\Debu
Vetor na posição 0: 1
Vetor na posição 1: 2
Vetor na posição 2: 3
Vetor na posição 3: 4
Vetor na posição 4: 5


1       2       3       4       5

Vetor na posição 0: Agua
Vetor na posição 1: Terra
Vetor na posição 2: Fogo
Vetor na posição 3: Ar
Vetor na posição 4: Coracao


Agua    Terra   Fogo    Ar      Coracao
Process returned 0 (0x0)   execution time : 11.073 s
Press any key to continue.
```

STRUCTS, TEMPLATES, VETORES/MATRIZES

Resumo Templates

- Utilizados para criar funções ou classes genéricas.



Resumo Templates

```
1   #include <iostream>
2   using namespace std;
3   #include <conio2.h>
4   #define TAM 2
5
6   template <typename tipoGenerico>
7   void leitura (tipoGenerico mat[TAM][TAM], int linhaInicial) {
8       int linha, coluna;
9       for (linha=0; linha<TAM; linha++) {
10          for (coluna=0; coluna<TAM; coluna++) {
11              gotoxy(coluna*10 + 3, linha * 2 + linhaInicial);
12              cin>>mat[linha][coluna];
13          }
14      }
15  }
16
17  int main()
18  {
19      int mat[TAM][TAM];
20      leitura(mat, 2);
21
22      char matrizChar[TAM][TAM];
23      leitura(matrizChar, 7);
24
25      return 0;
26  }
```

Exemplo de Struct, Template e Vetor

Exemplo 1

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

template <int N, typename Tipo>
struct Vetor {
  Tipo v[N];
};

template <int N, typename Tipo>
void leitura (Vetor<N,Tipo> &x) {
  for (int i=0;i<N;i++)
     cin>>x.v[i];
  cout<<endl;
}

template <int N, typename Tipo>
void escrita (Vetor<N,Tipo> x) {
  for (int i=0;i<N;i++)
     cout<<x.v[i]<<"  ";
  cout<<endl;
}
```

Exemplo de Struct, Template e Vetor

Exemplo 1

```
template <int N, typename Tipo>
Tipo soma (Vetor<N,Tipo> x) {
  Tipo d = 0;
  for (int i=0;i<N;i++)
     d += x.v[i];
  return d;
}

int main() {
  Vetor <5,int> a;
  leitura(a);
  escrita(a);
  cout<<"Soma do Vetor (int): "<<soma(a)<<endl<<endl;

  Vetor <7,float> b;
  leitura (b);
  cout<<"Soma do Vetor (float): "<<soma(b);
  return 0;
}
```

Exemplo de Struct, Template e Matriz

Exemplo 2

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

template <int N, int M, typename Tipo>
struct Estrutura {
    Tipo mat[N][M];
};

template <int N, int M, typename Tipo>
void leitura (Estrutura<N,M,Tipo> &a) {
  int i, j;
  cout<<endl<<endl;
  for (i=0;i<N;i++)
    for (j=0;j<M;j++)
      cin>>a.mat[i][j];
}
```

```
template <int N, int M, typename Tipo>
void escrita (Estrutura<N,M,Tipo> a) {
  int i, j;
  cout<<endl<<endl;
  for (i=0;i<N;i++) {
    for (j=0;j<M;j++) {
      cout<<a.mat[i][j]<<"\t";
    }
    cout<<endl;
  }
}
```

Exemplo de Struct, Template e Matriz

```
template <int N, int M, typename Tipo>
Estrutura<N,M,Tipo> soma (Estrutura<N,M,Tipo> a,
Estrutura<N,M,Tipo> b) {
    int i, j;
    Estrutura<N,M,Tipo> resultado;
    for (i=0;i<N;i++)
       for (j=0;j<M;j++)
          resultado.mat[i][j] = a.mat[i][j] + b.mat[i][j];
    return resultado;
}
```

Exemplo 2

```
int main() {
    Estrutura<3,3,float> m1, m2;
    leitura(m1);
    leitura(m2);
    escrita(soma(m1,m2));

    Estrutura<2,2,int> n1, n2, res;
    leitura(n1);
    leitura(n2);
    res = soma(n1,n2);
    escrita(res);

    return 0;
}
```