INTRODUÇÃO ÀS LINGUAGENS C/C++

// A história de C

A linguagem C evoluiu de duas linguagens de programação anteriores, BCPL e B. BCPL foi desenvolvida em 1967 por Martin Richards, como uma linguagem pra escrever software de sistemas operacionais e compiladores. Ken Thompson modelou muitas características de sua linguagem B inspirando-se em suas correspondentes em BCPL e usou B para criar as primeiras versões do sistema operacional UNIX no Bell Laboratories, em 1970, em um computador DEC PDP-7. Tanto BCPL como B eram linguagens *typeless*, ou seja, sem definição de tipos de dados – todo item de dados ocupava uma "palavra" na memória e o trabalho de tratar um item de dados como um número inteiro ou um número real, por exemplo, era de responsabilidade do programador.

A linguagem C foi derivada de B por Dennis Ritchie no Bell Laboratories e foi originalmente implementada em um computador DEC PDP-11 em 1972. C usa muitos conceitos importantes de BCPL e B, ao mesmo tempo que acrescenta tipos de dados e outras características. C se tornou inicialmente conhecida como a linguagem de desenvolvimento do sistema operacional UNIX. Hoje em dia, a maioria dos sistemas operacionais são escritos em C e/ou C++. C está atualmente disponível para a maioria dos computadores. C é independente de hardware. Com um projeto cuidadoso, é possível se escrever programas em C que são portáveis para a maioria dos computadores.

O uso difundido de C com vários tipos de computadores (às vezes chamados de *plataformas de hardware*) infelizmente levou a muitas variações da linguagem. Estas eram semelhantes, mas freqüentemente incompatíveis. Isto era um problema sério para desenvolvedores de programas que precisavam escrever programas portáveis que seriam executados em várias plataformas. Tornou-se claro que uma versão padrão de C era necessária. Em 1983, foi criado o comitê técnico X3J11 do *American National Standards Committee on Computers and Information Processing* (X3) para "produzir uma definição não-ambígua e independente de máquina da linguagem". Em 1989, o padrão foi aprovado. O ANSI cooperou com a International Standards Organization (ISO) para padronizar C a nível mundial; o documento de padronização conjunta foi publicado em 1990 e é chamado de ANSI/ISO 9899: 1990. Cópias deste documento podem ser pedidas ao ANSI.

// A história de C++

C++, uma extensão de C, foi desenvolvida por Bjarne Stroustrup no início dos anos 1980s no Bell Laboratories. C++ apresenta várias características que melhoram a linguagem C, mas o mais importante é que fornece recursos para a *programação orientada a objetos*.

Existe uma revolução em andamento na comunidade de software. Construir software rápida, correta e economicamente permanece um objetivo difícil de se atingir, e isto em uma época em que a demanda por softwares novos e mais poderosos está aumentando rapidamente. *Objetos* são, essencialmente, *componentes* de software reutilizáveis que modelam coisas do mundo real. Os desenvolvedores de software estão descobrindo que usar uma abordagem de implementação e projeto modulares, orientada a objetos, pode tornar os grupos de desenvolvimento de software muito mais produtivos do que é possível usando-se técnicas de programação anteriormente populares, como a programação estruturada. Os programas orientados a objetos são mais fáceis de entender, corrigir e modificar.

Muitas outras linguagens orientadas a objetos foram desenvolvidas, incluindo Smalltalk, desenvolvida no Palo Alto Research Center da Xerox (PARC). Smalltalk é uma linguagem orientada a objetos pura – literalmente, tudo nela é um objeto. C++ é uma linguagem híbrida – é possível programar em C++ tanto em um estilo semelhante ao de C, como em um estilo orientado a objetos, ou ambos.

(Retirado do livro: C++ Como programar, de Deitel & Deitel, Bookman)

```
// O primeiro programa em C (primeiro.c)
#include <stdio.h>

void main(void)
{
   printf("Programação C/C++");
}
// O primeiro programa em C++ (primeiro.cpp)
#include <iostream>

int main()
{
   std::cout << "Programação C/C++";

   return 0;
}
// Comentário (comentario1.c)
// Programa: COMENTARIO1.C
// Escrito por: xxx
// Data de criação: 01-01-2011
// Propósito: Ilustrar o uso de comentários em um programa C

#include <stdio.h>

void main(void)
{
   printf ("Programação C/C++");    // Exibe uma mensagem
}
// Outro tipo de comentário (comentario2 .c)
/* Programa: COMENTARIO2.C
   Escrito por: xxx
   Data de criação: 01-01-2011
   Propósito: Ilustrar o uso de comentários em um programa C
*/

#include <stdio.h>

void main(void)
{
   printf ("Programação C/C++");  /* Exibe uma mensagem */
}
// O cabeçalho
```

A partir de agora os programas nesta apostila não mais apresentarão cabeçalho por questão de espaço. Porém, é imprescindível que seus programas contenham NOME, FINALIDADE, AUTOR, DATAS, EMPRESA etc. no cabeçalho. Acostumem-se a colocá-lo em todos os seus programas.

// Legibilidade do seu programa (legibilidade.c)

```
#include <stdio.h>
void main(void){printf("Programação C/C++");}
```

// O ponto e vírgula

O ponto-e-vírgula é usado para separar um comando do outro. À medida que seus programas vão se tornando mais complexos, poderá acontecer de um comando não caber em uma única linha. Quando o compilador C examinar seu programa, ele procurará o ponto-e-vírgula para distinguir um comando do próximo.

// Caracteres de escape

Escape	Controle/Caracter
\0	nulo (null)
\a	campainha (bell)
\b	retrocesso (backspace)
\f	alimentacao de folha (form feed)
\n	nova linha (new line)
\r	retorno de carro (carriage return)
\t	tabulacao horizontal
\v	tabulacao vertical
\"	aspas (")
\'	apostrofo (')
\?	interrogacao (?)
\\	barra invertida (\)

// Imprimindo múltiplas linhas e tabulação (escape.cpp)

```
#include <iostream>

int main()
{
  std::cout << "Programação\n\tC/C++\n\n";
  return 0;
}
```

// Tipos de dados simples

Tipo	Tamanho	Uso
char	1 byte	número pequeno ou caracter ASCII
int	4 bytes	inteiro
float	4 bytes	real (precisão de 7 dígitos)
double	8 bytes	científico (precisão de 15 dígitos)

// Utilizando variáveis inteiras e atribuindo valores (variaveis.c)

```
void main(void)
{
  int idade;     // a idade do usuário em anos
  int peso;      // o peso do usuário em Kg
  int altura;    // a altura do usuário em centímetros

  idade = 41;    // atribui a idade ao usuário
  peso = 80;     // atribui o peso
  altura = 182;  // atribui a altura
}
```

```
// Declarando múltiplas variáveis
void main(void)
{
  int idade, peso, altura;
  float salario, impostos;
}
// Múltiplas atribuições de valores
  idade = peso = altura = 0; // apesar de permitido deve ser evitado

// Variáveis válidas e inválidas
```

Válidas	idade	soma37	multa_de_transito
Inválidas	4total	manda!ver	tipo-inteiro

```
// Atribuindo valores às variáveis em ponto flutuante
  raio = 123.45;
  raio = 1.2345E2;
// Formatando a entrada e a saída em C
```

Tipo	caracter de conversão de tipo (Requerido)
d	inteiro decimal
o	inteiro octal
x	inteiro hexadecimal
f	ponto flutuante: [-]dddd.dddd.
e	ponto flutuante com expoente: [-]d.dddde[+/-]ddd
c	caracter simples
s	string
l	long

```
// Imprimindo um valor inteiro em decimal, octal e hexadecimal (inteiro.c)
#include <stdio.h>

void main(void)
{
  int valor = 255;

  printf("O valor decimal %d em octal é %o\n", valor, valor);
  printf("O valor decimal %d em hexadecimal é %x\n", valor, valor);
  printf("O valor decimal %d em hexadecimal é %X\n", valor, valor);
}
// Exibindo valores do tipo float comum e em linguagem de engenharia (real.c)
#include <stdio.h>

void main(void)
{
  float preco = 525.75;
  float imposto = 0.06;

  printf("O custo do item é %f\n", preco);
  printf("E em linguagem de engenharia é %e\n", preco);
  printf("O imposto sobre a venda do item é %f\n", preco * imposto);
}
```

```
// Exibindo valores do tipo char usando printf() (caracter.c)
#include <stdio.h>

void main(void)
{
  printf("A letra é %c\n", 'A');
  printf("A letra é %c\n", 65);
}
// Exibindo uma string usando printf() (string.c)
#include <stdio.h>

void main(void)
{
  char titulo[255] = "Programação C/C++";

  printf("O nome desta apostila é %s\n", titulo);
}
```

// Modificadores para impressão

Depois do sinal %, seguem-se alguns modificadores, cuja sintaxe é a seguinte:

% [flag] [tamanho] [.precisão] tipo

[flag]	justificação de saída: (Opcional)
-	justificação à esquerda.
+	conversão de sinal (saída sempre com sinal: + ou -)

[tamanho]	especificação de tamanho (Opcional)
n	pelo menos n dígitos serão impressos (dígitos faltantes serão completados por brancos).
0n	pelo menos n dígitos serão impressos (dígitos faltantes serão completados por zeros).

[.precisão]	especificador de precisão, dígitos a direita do ponto decimal. (Opcional)
(nada)	padrão: 6 dígitos para reais.
.n	são impressos n dígitos decimais.

```
// Formatando um valor inteiro e preenchendo com zeros (tamanho.c)
#include <stdio.h>

void main(void)
{
  int valor = 5;

  printf ("%3d\n", valor);
  printf ("%03d\n", valor);
  printf ("%5d\n", valor);
  printf ("%05d\n", valor);
}
```

```
// Casas decimais e justificando à esquerda a saída de printf() (justifica.c)
#include <stdio.h>

void main(void)
{
  int int_valor = 5;
  float flt_valor = 3.33;

  printf("Justificado à direita %5d valor\n", int_valor);
  printf("Justificado à esquerda %-5d valor\n", int_valor);
  printf("Justificado à direita %7.2f valor\n", flt_valor);
  printf("Justificado à esquerda %-7.2f valor\n", flt_valor);
}
// Operações matemáticas básicas (matematica.c)
#include <stdio.h>

void main(void)
{
  int segundos_na_hora;
  float media;

  segundos_na_hora = 60 * 60;
  media = (5 + 10 + 15 + 20) / 4;
  printf("Número de segundos em uma hora %d\n", segundos_na_hora);
  printf("A média de 5, 10, 15 e 20 é %f\n", media);
  printf("48 minutos possui %d segundos\n", segundos_na_hora – 12*60);
}
// Divisão de inteiros
```

ATENÇÃO: a divisão de inteiros por inteiro será sempre inteira.

```
// Utilizando o módulo (modulo.c)
#include <stdio.h>

void main(void)
{
  int resto, resultado;

  resultado = 10 / 3;
  resto = 10 % 3;
  printf("10 dividido por 3 é %d \nResto é %d\n", resultado, resto);
}
// ++ e --
x = x + 1 → x++
x = x – 1 → x--
```

```
// Operador de incremento e decremento (incremento.c)
#include <stdio.h>

void main(void)
{
  int valor = 10;

  printf("Usando sufixo %d\n", valor++);
  printf("Valor após o incremento %d\n", valor);
  printf("Usando prefixo %d\n", ++valor);
  printf("Valor após o incremento %d\n", valor);

  printf("Usando postfixo %d\n", valor--);
  printf("Valor após o decremento %d\n", valor);
  printf("Usando prefixo %d\n", --valor);
  printf("Valor após o decremento %d\n", valor);
}
// Lendo e somando dois inteiros em C (leitura.c)
#include <stdio.h>

void main(void)
{
  int inteiro1, inteiro2, soma;

  printf("Digite o primeiro número\n");
  scanf("%d", &inteiro1);
  printf("Digite o segundo número \n";
  scanf("%d", &inteiro2);
  soma = inteiro1 + inteiro2;
  printf("A soma é %d\n", soma);
}
// Lendo e somando dois inteiros em C++ (leitura.cpp)
#include <iostream>

int main()
{
  int inteiro1, inteiro2, soma;

  std::cout << "Digite o primeiro número\n";
  std::cin >> inteiro1;
  std::cout << " Digite o segundo número \n";
  std::cin >> inteiro2;
  soma = inteiro1 + inteiro2;
  std::cout << "A soma é " << soma << std::endl;
  return 0;
}
```

```
// Utilizando using (using.cpp)
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

int main()
{
  int inteiro1, inteiro2, soma;

  cout << "Digite o primeiro número\n";
  cin >> inteiro1;
  cout << " Digite o segundo número \n";
  cin >> inteiro2;
  soma = inteiro1 + inteiro2;
  cout << "A soma é " << soma << endl;
  return 0;
}
// Utilizando using namespace (namespace.cpp)
#include <iostream>
using namespace std;

int main()
{
  int inteiro1, inteiro2, soma;

  cout << "Digite o primeiro número\n";
  cin >> inteiro1;
  cout << " Digite o segundo número \n";
  cin >> inteiro2;
  soma = inteiro1 + inteiro2;
  cout << "A soma é " << soma << endl;
  return 0;
}
```