Programação Orientada a Objetos

Evolução do C para o C++: Entrada/Saída de Dados

1. Introdução

A linguagem C é uma das mais conhecidas por seu controle preciso sobre o hardware e pela manipulação de memória com **malloc/free** e **ponteiros**. O C++, uma extensão do C, traz melhorias significativas com **classes**, **encapsulamento**, **herança** e um controle mais seguro e eficiente de memória com **new/delete**. Além disso, o C++ introduz um sistema de entrada e saída mais seguro e legível com os operadores **cout** e **cin**, substituindo os antigos **printf** e **scanf** do C.

---

2. Entrada e Saída de Dados: printf/scanf (C) vs cout/cin (C++)

2.1. Entrada e Saída em C

No C, usamos as funções **printf()** e **scanf()** para realizar a entrada e saída de dados.

Exemplo de Entrada e Saída em C

```c
#include <stdio.h>

int main() {
    int idade;
    float altura;

    printf("Digite a sua idade: ");
    scanf("%d", &idade);

    printf("Digite a sua altura (em metros): ");
    scanf("%f", &altura);

    printf("Você tem %d anos e %.2f metros de altura.\n", idade, altura);

    return 0;
}
```

- **printf**: Exibe texto formatado na tela.
- **scanf**: Lê dados da entrada padrão e armazena nas variáveis.
- O formato `%d` é para inteiros, `%f` para números de ponto flutuante.
- **&** (operador de endereço) é necessário no **scanf** para passar o endereço da variável.

2.2. Entrada e Saída em C++

No C++, o **cout** e o **cin** substituem **printf** e **scanf**. Eles são mais seguros, detectam automaticamente o tipo de dados e permitem uma sintaxe mais limpa e legível.

Exemplo de Entrada e Saída em C++

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

int main() {
    int idade;
    float altura;

    cout << "Digite a sua idade: ";
    cin >> idade;

    cout << "Digite a sua altura (em metros): ";
    cin >> altura;

    cout << "Você tem " << idade << " anos e " << altura << " metros de altura."
<< endl;

    return 0;
}
```

- **cout**: Saída de texto, semelhante a **printf**.
- **cin**: Entrada de dados, semelhante a **scanf**.
- **<<**: Envia texto/dados para a saída.
- **>>**: Captura os dados de entrada e armazena nas variáveis.
- Não precisa de operador de endereço **&**, pois o C++ sabe onde armazenar os dados automaticamente.
- O **endl** (end line) quebra a linha, semelhante a `\n` no **printf**.

---

3. Formatação da Saída

A formatação é importante para exibir os dados de forma legível. Vamos ver como isso é feito no C com **printf** e no C++ com **cout**.

3.1. Formatação no C com printf

A função **printf()** usa especificadores de formato e controladores de largura, precisão e alinhamento.

Principais Especificadores de Formato

Especificador	Significado
`%d`	Inteiro decimal
`%f`	Float (ponto flutuante)

Especificador	Significado
`%c`	Caractere
`%s`	String (cadeia de caracteres)
`%x` / `%X`	Hexadecimal
`%o`	Octal

Modificadores de Largura e Precisão

- `%5d`: Inteiro com largura de 5 posições.
- `%08d`: Inteiro com zeros à esquerda.
- `%.2f`: Float com 2 casas decimais.
- `%10s`: String alinhada à direita em 10 posições.

Exemplo de Formatação em C

```c
#include <stdio.h>

int main() {
    int x = 42;
    float pi = 3.14159;

    printf("Inteiro: %5d\n", x);  // Largura de 5
    printf("Float (2 decimais): %.2f\n", pi); // Duas casas decimais
    printf("Float (6 decimais): %.6f\n", pi); // Seis casas decimais
    printf("Com zeros: %08d\n", x);  // Completa com zeros à esquerda

    return 0;
}
```

3.2. Formatação no C++ com cout

No C++, utilizamos manipuladores de formato, como **setw**, **setprecision**, **fixed**, **scientific** e **left/right**.

Principais Manipuladores

Manipulador	Função
`setw(n)`	Define a largura do campo (n posições)
`setprecision(n)`	Define a precisão para números de ponto flutuante
`fixed`	Usa notação decimal (padrão)
`scientific`	Usa notação científica (1.23e+05)
`left` / `right`	Alinha à esquerda ou à direita

> Para usar esses manipuladores, é necessário incluir a biblioteca **iomanip**.

Exemplo de Formatação em C++

```cpp
#include <iostream>
#include <iomanip> // Biblioteca necessária
using namespace std;

int main() {
    int x = 42;
    float pi = 3.14159;

    cout << "Inteiro: " << setw(5) << x << endl; // Largura de 5
    cout << "Float (2 decimais): " << fixed << setprecision(2) << pi << endl; // 2 casas decimais
    cout << "Float (6 decimais): " << fixed << setprecision(6) << pi << endl; // 6 casas decimais
    cout << "Com zeros: " << setw(8) << setfill('0') << x << endl; // Completa com zeros à esquerda

    return 0;
}
```

Comparação: printf vs cout

Função	C (printf)	C++ (cout)
Largura	`%5d`	`setw(5)`
Casas Decimais	`%.2f`	`setprecision(2)` + `fixed`
Zeros à Esquerda	`%08d`	`setw(8)` + `setfill('0')`
Quebra de Linha	`\n`	`endl` ou `\n`

---

4. Diferenças Fundamentais (printf/scanf vs cout/cin)

Aspecto	printf/scanf (C)	cout/cin (C++)
Sintaxe	Usa especificadores (%d, %f)	Usa operadores `<<` e `>>`
Detecta o Tipo	Não (precisa %d, %f, etc.)	Detecta automaticamente
Alinhamento	%5d (direita)	`setw(5)` e `left` / `right`
Casas Decimais	%.2f (com float)	`setprecision(2)` + `fixed`
Necessário & no scanf?	Sim	Não

Aspecto	printf/scanf (C)	cout/cin (C++)
Modificadores de Largura	`%10s`, `%08d`	`setw(10)`, `setfill('0')`

- **C**: Requer maior controle manual da entrada, saída e memória.
- **C++**: Traz mais segurança e expressividade com **cout**, **cin**, **new**, **delete** e **classes**.
- **cout/cin**: Facilita a entrada e saída, sem necessidade de especificadores como `%d` e `%f`.

Exercícios

Exercício 1: Cálculo de Área de um Círculo

Objetivo: Implementar um programa que leia o raio de um círculo e exiba a área com 2 e 6 casas decimais.

Fórmula: (A = \pi \cdot r^2)

Versão em C

Requisitos:

1. Solicitar que o usuário insira o raio (float).
2. Calcular a área do círculo.
3. Exibir o resultado com 2 e 6 casas decimais.

Versão em C++

Requisitos:

1. Usar `cin` para entrada de dados.
2. Exibir a área com 2 e 6 casas decimais utilizando `fixed` e `setprecision`.

Exercício 2: Formatação de Saída de Números

Objetivo: Implementar um programa que leia 3 inteiros e exiba-os de forma alinhada à direita, ocupando 5 posições para cada número.

Versão em C

Requisitos:

1. Solicitar 3 números inteiros.
2. Exibir cada número em uma linha ocupando **5 posições de largura**.

Versão em C++

Requisitos:

1. Usar `cin` para entrada de dados.
2. Usar `setw(5)` para formatar cada número.

Exercício 3: Cadastro de Informações Pessoais

Objetivo: Criar um programa que leia **nome, idade e altura** de uma pessoa e exiba os dados formatados.

Versão em C

Requisitos:

1. Solicitar nome, idade e altura.
2. Exibir os dados no formato: "Nome: [nome], Idade: [idade] anos, Altura: [altura] m".

Versão em C++

Requisitos:

1. Usar `cin` para entrada de dados.
2. Usar `cout` para exibir as informações de forma clara.
