0809-Programação em C/C++ - fundamentos

Sistema de Aprendizagem (APZ)

Linguagens de Programação - Programação Web - FAD

|  |  |
| --- | --- |
| **FORMADOR/A** | **DATA** |
| Luis Cardoso | agosto de 2024 |

**Índice**

[20240807---tarde 2](#_Toc173938402)

[Estrutura de um programa em C++ 2](#_Toc173938403)

[Variáveis e Constantes 7](#_Toc173938404)

[20240807---tarde 12](#_Toc173938405)

[COUT 12](#_Toc173938406)

[CIN 18](#_Toc173938407)

[Diretiva #define 28](#_Toc173938408)

20240807---tarde

Estrutura de um programa em C++

C++ é uma linguagem de programação poderosa e versátil, amplamente usada em diversas áreas, desde sistemas operacionais até jogos. A estrutura básica de um programa em C++ é composta por vários componentes essenciais. Vamos explorar cada um deles com exemplos completos.

**Estrutura Básica de um Programa em C++**

1. **Diretivas de Pré-processador**
2. **Função Principal (main)**
3. **Declaração de Variáveis**
4. **Estruturas de Controle**
5. **Funções**

**1. Diretivas de Pré-processador**

Essas diretivas instruem o compilador a realizar algumas ações antes da compilação do código. A mais comum é #include, que inclui bibliotecas.

#include <iostream> // Inclui a biblioteca de entrada e saída padrão

#include <cmath> // Inclui a biblioteca matemática

**2. Função Principal (main)**

Todo programa em C++ começa a execução a partir da função main.

int main() {

// Corpo da função principal

return 0;

}

**3. Declaração de Variáveis**

Variáveis são declaradas para armazenar dados.

int idade = 25;

double salario = 4500.50;

char letra = 'A';

**4. Estruturas de Controle**

Estruturas de controle são usadas para tomar decisões e repetir ações.

**Condicional (if-else)**

if (idade > 18) {

std::cout << "Maior de idade" << std::endl;

} else {

std::cout << "Menor de idade" << std::endl;

}

Loop (for)

for (int i = 0; i < 5; i++) {

std::cout << "Contagem: " << i << std::endl;

}

for (int i = 0; i < 5; i++) {

std::cout << "Contagem: " << i << std::endl;

}

**5. Funções**

Funções são blocos de código reutilizáveis.

// Declaração da função

int soma(int a, int b) {

return a + b;

}

int main() {

int resultado = soma(5, 3);

std::cout << "Resultado da soma: " << resultado << std::endl;

return 0;

}

**Exemplo Completo de um Programa em C++**

Vamos juntar todos esses componentes em um programa completo.

#include <iostream> // Biblioteca de entrada e saída

// Declaração de função

void saudacao() {

std::cout << "Olá, bem-vindo ao programa!" << std::endl;

}

int soma(int a, int b) {

return a + b;

}

int main() {

saudacao(); // Chamada da função saudacao

int x, y;

std::cout << "Digite dois números: ";

std::cin >> x >> y;

int resultado = soma(x, y);

std::cout << "A soma dos números é: " << resultado << std::endl;

if (resultado > 10) {

std::cout << "A soma é maior que 10" << std::endl;

} else {

std::cout << "A soma é 10 ou menor" << std::endl;

}

for (int i = 0; i < 3; i++) {

std::cout << "Iteração: " << i << std::endl;

}

return 0;

}

**Explicação do Programa Completo**

1. **Diretivas de Pré-processador**

#include <iostream>

 Inclui a biblioteca de entrada e saída.

 **Funções**

void saudacao() {

std::cout << "Olá, bem-vindo ao programa!" << std::endl;

}

int soma(int a, int b) {

return a + b;

}

 Declara duas funções: uma para exibir uma saudação e outra para somar dois números.

 **Função Principal**

int main() {

saudacao(); // Chamada da função saudacao

int x, y;

std::cout << "Digite dois números: ";

std::cin >> x >> y;

int resultado = soma(x, y);

std::cout << "A soma dos números é: " << resultado << std::endl;

if (resultado > 10) {

std::cout << "A soma é maior que 10" << std::endl;

} else {

std::cout << "A soma é 10 ou menor" << std::endl;

}

for (int i = 0; i < 3; i++) {

std::cout << "Iteração: " << i << std::endl;

}

return 0;

}

 **Chamada de Funções**: A função saudacao é chamada para exibir uma mensagem de boas-vindas.

 **Entrada de Usuário**: Lê dois números do usuário.

 **Cálculo e Saída**: Chama a função soma para calcular a soma dos números e exibe o resultado.

 **Estrutura Condicional**: Verifica se o resultado da soma é maior que 10.

 **Loop**: Executa um loop for para exibir iterações.

Variáveis e Constantes

**Variáveis**

Variáveis são usadas para armazenar dados que podem mudar durante a execução do programa.

#include <iostream>

using namespace std;

int main() {

int idade = 25;

double salario = 4500.50;

char letra = 'A';

cout << "Idade: " << idade << endl;

cout << "Salário: " << salario << endl;

cout << "Letra: " << letra << endl;

return 0;

}

**Constantes**

Constantes são usadas para armazenar dados que não podem ser alterados durante a execução do programa.

#include <iostream>

using namespace std;

int main() {

const int DIAS\_NO\_ANO = 365;

const double PI = 3.14159;

cout << "Dias no ano: " << DIAS\_NO\_ANO << endl;

cout << "Valor de PI: " << PI << endl;

return 0;

}

**Tipos de Dados**

C++ possui vários tipos de dados, incluindo inteiros, caracteres, booleanos e ponto flutuante.

**Tipos de Dados Inteiros**

#include <iostream>

using namespace std;

int main() {

int inteiro = 100;

unsigned int inteiroPositivo = 150;

cout << "Inteiro: " << inteiro << endl;

cout << "Inteiro Positivo: " << inteiroPositivo << endl;

return 0;

}

Tipos de Dados Caracteres

#include <iostream>

using namespace std;

int main() {

char letra = 'A';

cout << "Letra: " << letra << endl;

return 0;

}

Tipos de Dados Booleanos

#include <iostream>

using namespace std;

int main() {

bool verdadeiro = true;

bool falso = false;

cout << "Verdadeiro: " << verdadeiro << endl; // Saída: 1

cout << "Falso: " << falso << endl; // Saída: 0

return 0;

}

**Variáveis de Ponto Flutuante**

Para armazenar números com casas decimais, usamos os tipos float, double e long double.

#include <iostream>

using namespace std;

int main() {

float salarioMensal = 4500.75f;

double salarioAnual = 54009.30;

long double grandeNumero = 1.2345678901234567890L;

cout << "Salário Mensal: " << salarioMensal << endl;

cout << "Salário Anual: " << salarioAnual << endl;

cout << "Grande Número: " << grandeNumero << endl;

return 0;

}

**Exemplo Completo com Variáveis e Constantes**

Vamos criar um programa completo que demonstra o uso de variáveis, constantes e tipos de dados.

#include <iostream>

using namespace std;

int main() {

// Variáveis

int idade = 25;

double salarioMensal = 4500.50;

char letraInicial = 'A';

bool empregado = true;

// Constantes

const double PI = 3.14159;

const int DIAS\_NO\_ANO = 365;

// Saída de variáveis

cout << "Idade: " << idade << endl;

cout << "Salário Mensal: " << salarioMensal << endl;

cout << "Letra Inicial: " << letraInicial << endl;

cout << "Empregado: " << empregado << endl;

// Saída de constantes

cout << "Valor de PI: " << PI << endl;

cout << "Dias no Ano: " << DIAS\_NO\_ANO << endl;

// Operações com variáveis de ponto flutuante

double raio = 2.5;

double areaCirculo = PI \* raio \* raio;

cout << "Área do Círculo com raio " << raio << " é: " << areaCirculo << endl;

return 0;

}

**Explicação do Exemplo Completo**

1. **Declaração de Variáveis**: idade, salarioMensal, letraInicial e empregado são variáveis que armazenam dados diferentes.
2. **Declaração de Constantes**: PI e DIAS\_NO\_ANO são constantes que armazenam valores fixos.
3. **Saída de Variáveis e Constantes**: cout é usado para exibir os valores das variáveis e constantes.
4. **Operações com Variáveis de Ponto Flutuante**: Calculamos a área de um círculo usando a constante PI e uma variável raio.

Este exemplo mostra como trabalhar com variáveis, constantes e diferentes tipos de dados em C++.

20240807---tarde

20240808----Manhã

Atividade1-idade-exata-utilizador

Vamos criar um exercício em C++ que calcula a idade exata de um usuário com base na data de nascimento e na data atual. Para simplificar o exercício, assumiremos que as datas são fornecidas no formato dia/mês/ano.

**Exercício**

**Descrição:** Escreva um programa em C++ que solicite ao usuário a data de nascimento e a data atual. O programa deve então calcular e exibir a idade exata do usuário em anos, meses e dias.

**Requisitos:**

1. Solicitar a data de nascimento do usuário no formato dia/mês/ano.
2. Solicitar a data atual no mesmo formato.
3. Calcular a idade exata do usuário.
4. Exibir a idade em anos, meses e dias.

**Exemplo de Entrada e Saída**

**Entrada:**

Digite sua data de nascimento (dia mes ano): 15 8 1990

Digite a data atual (dia mes ano): 7 8 2024

Saída:

#include <iostream>

using namespace std;

struct Data {

int dia;

int mes;

int ano;

};

bool isAnoBissexto(int ano) {

if (ano % 4 == 0) {

if (ano % 100 == 0) {

if (ano % 400 == 0)

return true;

else

return false;

}

else

return true;

}

else

return false;

}

int diasNoMes(int mes, int ano) {

switch (mes) {

case 1: case 3: case 5: case 7: case 8: case 10: case 12:

return 31;

case 4: case 6: case 9: case 11:

return 30;

case 2:

return isAnoBissexto(ano) ? 29 : 28;

default:

return 0;

}

}

void calcularIdade(Data nascimento, Data atual) {

int anos = atual.ano - nascimento.ano;

int meses = 0;

int dias = 0;

if (atual.mes < nascimento.mes || (atual.mes == nascimento.mes && atual.dia < nascimento.dia)) {

anos--;

}

if (atual.mes >= nascimento.mes) {

meses = atual.mes - nascimento.mes;

} else {

meses = 12 - (nascimento.mes - atual.mes);

}

if (atual.dia >= nascimento.dia) {

dias = atual.dia - nascimento.dia;

} else {

if (meses == 0) {

anos--;

meses = 11;

} else {

meses--;

}

dias = diasNoMes((atual.mes == 1) ? 12 : (atual.mes - 1), atual.ano) - nascimento.dia + atual.dia;

}

cout << "Você tem " << anos << " anos, " << meses << " meses e " << dias << " dias." << endl;

}

int main() {

Data nascimento, atual;

cout << "Digite sua data de nascimento (dia mes ano): ";

cin >> nascimento.dia >> nascimento.mes >> nascimento.ano;

cout << "Digite a data atual (dia mes ano): ";

cin >> atual.dia >> atual.mes >> atual.ano;

calcularIdade(nascimento, atual);

return 0;

}

**Explicação do Código**

1. **Estrutura Data**:
   * Definimos uma estrutura Data para armazenar dia, mês e ano.
2. **Função isAnoBissexto**:
   * Determina se um ano é bissexto.
3. **Função diasNoMes**:
   * Retorna o número de dias em um determinado mês e ano, levando em consideração anos bissextos.
4. **Função calcularIdade**:
   * Calcula a idade exata em anos, meses e dias, com base nas datas de nascimento e atual fornecidas.
   * Ajusta os cálculos de anos, meses e dias para considerar os casos em que a data atual está antes da data de nascimento no mês ou dia.
5. **Função main**:
   * Solicita ao usuário a data de nascimento e a data atual.
   * Chama a função calcularIdade para calcular e exibir a idade exata do usuário.

Solução simplificada

Vamos criar uma solução mais simples para calcular a idade de um usuário com base na data de nascimento e na data atual, sem considerar os dias exatos, apenas a idade em anos e meses.

**Código em C++**

#include <iostream>

using namespace std;

struct Data {

int dia;

int mes;

int ano;

};

void calcularIdade(Data nascimento, Data atual) {

int anos = atual.ano - nascimento.ano;

int meses = atual.mes - nascimento.mes;

int dias = atual.dia - nascimento.dia;

if (dias < 0) {

meses--;

dias += 30; // Aproximadamente, assume-se 30 dias em um mês

}

if (meses < 0) {

anos--;

meses += 12;

}

cout << "Você tem " << anos << " anos, " << meses << " meses e " << dias << " dias." << endl;

}

int main() {

Data nascimento, atual;

cout << "Digite sua data de nascimento (dia mes ano): ";

cin >> nascimento.dia >> nascimento.mes >> nascimento.ano;

cout << "Digite a data atual (dia mes ano): ";

cin >> atual.dia >> atual.mes >> atual.ano;

calcularIdade(nascimento, atual);

return 0;

}

**Explicação do Código**

1. **Estrutura Data**:
   * Definimos uma estrutura Data para armazenar dia, mês e ano.
2. **Função calcularIdade**:
   * Calcula a idade em anos, meses e dias de forma simplificada.
   * Subtrai os anos diretamente.
   * Calcula a diferença de meses. Se o mês atual for menor que o mês de nascimento, subtrai 1 ano e ajusta os meses.
   * Calcula a diferença de dias. Se o dia atual for menor que o dia de nascimento, subtrai 1 mês e ajusta os dias.
3. **Função main**:
   * Solicita ao usuário a data de nascimento e a data atual.
   * Chama a função calcularIdade para calcular e exibir a idade aproximada do usuário.

Esta solução simples não considera a precisão dos dias em meses específicos (e.g., fevereiro com 28 ou 29 dias), mas fornece uma aproximação razoável da idade em anos e meses.

Solução João Monteiro

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

int main() {

// Obtém o tempo atual do sistema

auto now = system\_clock::now();

// Converte para o tempo do calendário

auto now\_time\_t = system\_clock::to\_time\_t(now);

std::tm now\_tm = \*std::localtime(&now\_time\_t);

int dia\_nascimento;

int mes\_nascimento;

int ano\_nascimento;

// Data de nascimento fixada

cout << "Digite a o ano de nascimento: " << endl;

cin>> ano\_nascimento;

cout << "Digite a mes de nascimento: " << endl;

cin>> mes\_nascimento;

cout << "Digite a o dia de nascimento: "<< endl;

cin>> dia\_nascimento;

// Calcula a idade

int idade = now\_tm.tm\_year + 1900 - ano\_nascimento; // Idade base

// Ajusta a idade se a data de nascimento ainda não ocorreu neste ano

if (now\_tm.tm\_mon + 1 < mes\_nascimento ||

(now\_tm.tm\_mon + 1 == mes\_nascimento && now\_tm.tm\_mday < dia\_nascimento)) {

idade--;

}

cout << "Idade: " << idade << endl;

return 0;

}

Helena

*//Função 'main' p/ ler a data atual e a data de nascimento do usuário e void 'calcularIdade' p/ obter a idade detalhada;*

*//Primeiro defini uma estrutura p/armazenar uma data com dia, mes e ano com a declaração 'Data';*

*//Depois usei void CalcularIdade p/ calcular a diferença entre a data atual e a data de nascimento;*

*//Usei a função 'if < 0' p/ ajustar a contagem de dias e meses, quando o resultado for negativo;*

*//O programa exibe a idade em anos, meses e dias;*

*//E por último pede ao usuário p/ digitar 0 p/ sair.*

#include <iostream>

#include <locale>

using namespace std;

struct Data {

    int dia;

    int mes;

    int ano;

};

void calcularIdade(Data dataAtual, Data dataNascimento, int& anos, int& meses, int& dias) {

    anos = dataAtual.ano - dataNascimento.ano;

    meses = dataAtual.mes - dataNascimento.mes;

    dias = dataAtual.dia - dataNascimento.dia;

    if (dias < 0) {

        meses--;

        dias += 30; *// Aproximação simples p/ o número de dias no mês anterior*

    }

    if (meses < 0) {

        anos--;

        meses += 12;

    }

}

int main() {

    Data dataAtual;

    Data dataNascimento;

    cout << "Digite a data atual (dia mes ano): ";

    cin >> dataAtual.dia >> dataAtual.mes >> dataAtual.ano;

    cout << "Digite sua data de nascimento (dia mes ano): ";

    cin >> dataNascimento.dia >> dataNascimento.mes >> dataNascimento.ano;

    int anos, meses, dias;

    calcularIdade(dataAtual, dataNascimento, anos, meses, dias);

    cout << "Você tem " << anos << " anos, " << meses << " meses e " << dias << " dias." << endl;

*// Comando para sair*

    cout << "Digite 0 para sair: ";

    int sair;

    cin >> sair;

    return 0;

}

COUT

A função cout é usada em C++ para exibir informações na tela. Ela faz parte da biblioteca padrão de entrada e saída (iostream). Vamos explorar como usar cout e como formatar a saída para torná-la mais legível e organizada.

**Uso Básico de cout**

A função cout é usada junto com o operador de inserção (<<) para enviar dados ao fluxo de saída padrão (geralmente o console).

#include <iostream>

using namespace std;

int main() {

cout << "Hello, World!" << endl;

return 0;

}

Neste exemplo, cout exibe a mensagem "Hello, World!" no console, seguida por uma nova linha (endl).

**Exibindo Variáveis**

Podemos usar cout para exibir o valor de variáveis.

#include <iostream>

using namespace std;

int main() {

int idade = 25;

double salario = 4500.50;

char letra = 'A';

cout << "Idade: " << idade << endl;

cout << "Salário: " << salario << endl;

cout << "Letra: " << letra << endl;

return 0;

}

**Concatenando Saídas**

Podemos concatenar várias saídas em uma única linha.

#include <iostream>

using namespace std;

int main() {

int idade = 25;

double salario = 4500.50;

cout << "Idade: " << idade << ", Salário: " << salario << endl;

return 0;

}

**Formatação da Saída**

Para formatar a saída de forma mais controlada, podemos usar manipuladores de fluxo (stream manipulators) como setw, setprecision, fixed, e left/right.

**Incluindo a Biblioteca iomanip**

Para usar os manipuladores de fluxo, precisamos incluir a biblioteca iomanip.

#include <iostream>

#include <iomanip>

using namespace std;

**//Ajustando a Largura da Coluna (setw)**

//O manipulador setw ajusta a largura do campo de saída.

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int idade = 25;

double salario = 4500.50;

cout << setw(10) << "Idade" << setw(10) << "Salário" << endl;

cout << setw(10) << idade << setw(10) << salario << endl;

return 0;

}

**Definindo a Precisão de Pontos Flutuantes (setprecision)**

O manipulador setprecision define o número de dígitos após o ponto decimal.

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double pi = 3.14159265359;

cout << "Valor de PI com diferentes precisões:" << endl;

cout << setprecision(2) << pi << endl;

cout << setprecision(4) << pi << endl;

cout << setprecision(6) << pi << endl;

return 0;

}

**Usando fixed e showpoint**

Os manipuladores fixed e showpoint são usados para exibir números em ponto flutuante com um número fixo de casas decimais.

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double salario = 4500.50;

cout << fixed << showpoint;

cout << setprecision(2);

cout << "Salário: " << salario << endl;

return 0;

}

**Ajustando a Alinhamento (left e right)**

Os manipuladores left e right ajustam o alinhamento do texto.

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int idade = 25;

double salario = 4500.50;

cout << left << setw(10) << "Idade" << setw(10) << "Salário" << endl;

cout << left << setw(10) << idade << setw(10) << salario << endl;

return 0;

}

**Exemplo Completo de Formatação**

Vamos juntar todos esses conceitos em um exemplo completo.

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

// Dados

int idade = 25;

double salario = 4500.50;

double pi = 3.14159265359;

const int largura = 15;

// Cabeçalho

cout << left << setw(largura) << "Descrição" << setw(largura) << "Valor" << endl;

cout << setfill('-') << setw(2 \* largura) << "-" << setfill(' ') << endl;

// Saída formatada

cout << left << setw(largura) << "Idade" << right << setw(largura) << idade << endl;

cout << left << setw(largura) << "Salário" << right << setw(largura) << fixed << setprecision(2) << salario << endl;

cout << left << setw(largura) << "PI" << right << setw(largura) << fixed << setprecision(5) << pi << endl;

return 0;

}

**Explicação do Exemplo Completo**

1. **Incluindo Bibliotecas**: Incluímos iostream para operações de entrada e saída e iomanip para formatação.
2. **Definindo Dados**: Declaramos variáveis idade, salario, e pi, e uma constante largura para definir a largura da coluna.
3. **Cabeçalho**: Usamos setw e left para criar um cabeçalho alinhado à esquerda.
4. **Linha de Separação**: Usamos setfill para criar uma linha de separação.
5. **Saída Formatada**: Exibimos os dados formatados usando setw, left, right, fixed, e setprecision.

Este exemplo mostra como usar cout para exibir variáveis e como formatar a saída para criar uma apresentação mais organizada e legível.

CIN

A função cin em C++ é usada para ler dados de entrada do usuário. Ela também faz parte da biblioteca padrão de entrada e saída (iostream). Vamos explorar como usar cin e como formatar a entrada para garantir que os dados sejam lidos corretamente.

**Uso Básico de cin**

A função cin é usada junto com o operador de extração (>>) para ler dados do fluxo de entrada padrão (geralmente o console).

#include <iostream>

using namespace std;

int main() {

int idade;

cout << "Digite sua idade: ";

cin >> idade;

cout << "Sua idade é: " << idade << endl;

return 0;

}

Neste exemplo, cin lê um valor inteiro do usuário e o armazena na variável idade.

**Lendo Diferentes Tipos de Dados**

Podemos usar cin para ler diferentes tipos de dados, como int, double, char, e string.

#include <iostream>

using namespace std;

int main() {

int idade;

double salario;

char inicial;

string nome;

cout << "Digite sua idade: ";

cin >> idade;

cout << "Digite seu salário: ";

cin >> salario;

cout << "Digite sua inicial: ";

cin >> inicial;

cout << "Digite seu nome: ";

cin >> nome; // Note que isso só lê uma palavra (até o primeiro espaço)

cout << "Idade: " << idade << endl;

cout << "Salário: " << salario << endl;

cout << "Inicial: " << inicial << endl;

cout << "Nome: " << nome << endl;

return 0;

}

**Lendo Strings Com Espaços**

Para ler uma string que pode conter espaços, usamos a função getline.

#include <iostream>

#include <string>

using namespace std;

int main() {

string nomeCompleto;

cout << "Digite seu nome completo: ";

cin.ignore(); // Ignora o caractere de nova linha pendente

getline(cin, nomeCompleto);

cout << "Nome completo: " << nomeCompleto << endl;

return 0;

}

**#include <iostream>**

**#include <string>**

**using namespace std;**

**int main() {**

**string nomeCompleto;**

**for(int i=0;i<3;i++){**

**//cin.ignore;**

**cout << "Digite seu nome completo: ";**

**getline(cin, nomeCompleto);**

**cout << "Nome completo: " << nomeCompleto << endl;**

**}**

**return 0;**

**}**

**Tratamento de Erros e Limpeza de cin**

Quando cin encontra uma entrada inválida, ele entra em um estado de falha. Podemos tratar isso limpando o estado de erro e ignorando a entrada incorreta.

#include <iostream>

#include <limits> // Necessário para numeric\_limits

using namespace std;

int main() {

int idade;

cout << "Digite sua idade: ";

cin >> idade;

// Verifica se a entrada falhou

if (cin.fail()) {

cin.clear(); // Limpa o estado de erro

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // Ignora a entrada incorreta

cout << "Entrada inválida. Por favor, digite um número inteiro." << endl;

} else {

cout << "Sua idade é: " << idade << endl;

}

return 0;

}**Exemplo Completo com cin e Tratamento de Erros**

Vamos criar um exemplo completo que usa cin para ler diferentes tipos de dados e trata possíveis erros de entrada.

#include <iostream>

#include <iomanip>

#include <limits>

using namespace std;

int main() {

int idade;

double salario;

char inicial;

string nomeCompleto;

// Lendo idade

cout << "Digite sua idade: ";

cin >> idade;

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Entrada inválida para idade. Por favor, digite um número inteiro." << endl;

return 1;

}

// Lendo salário

cout << "Digite seu salário: ";

cin >> salario;

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Entrada inválida para salário. Por favor, digite um número decimal." << endl;

return 1;

}

// Lendo inicial

cout << "Digite sua inicial: ";

cin >> inicial;

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Entrada inválida para inicial. Por favor, digite um único caractere." << endl;

return 1;

}

// Lendo nome completo

cout << "Digite seu nome completo: ";

cin.ignore(); // Ignora o caractere de nova linha pendente

getline(cin, nomeCompleto);

// Exibindo os dados formatados

cout << fixed << showpoint;

cout << setprecision(2);

cout << left << setw(15) << "Idade" << ": " << idade << endl;

cout << left << setw(15) << "Salário" << ": " << salario << endl;

cout << left << setw(15) << "Inicial" << ": " << inicial << endl;

cout << left << setw(15) << "Nome Completo" << ": " << nomeCompleto << endl;

return 0;

}

**Explicação do Exemplo Completo**

1. **Incluindo Bibliotecas**: Incluímos iostream para operações de entrada e saída, iomanip para formatação e limits para manipulação de limites de fluxo.
2. **Lendo Dados**: Lemos idade, salario, inicial e nomeCompleto do usuário.
3. **Tratamento de Erros**: Usamos cin.fail() para verificar se a entrada foi inválida. Se for, limpamos o estado de erro e ignoramos a entrada incorreta.
4. **Exibindo Dados Formatados**: Usamos manipuladores de fluxo (setw, fixed, showpoint, setprecision) para exibir os dados de forma formatada.

Este exemplo mostra como usar cin para ler dados de entrada do usuário e como tratar e formatar essa entrada para garantir que os dados sejam corretos e bem apresentados.

**Operador de Endereço (&)**

O operador de endereço (&) é usado para obter o endereço de memória de uma variável. Isso é particularmente útil quando se trabalha com ponteiros.

**Exemplos Básicos**

#include <iostream>

using namespace std;

int main() {

int x = 10;

int \*p = &x; // p é um ponteiro que armazena o endereço de x

cout << "Valor de x: " << x << endl;

cout << "Endereço de x: " << &x << endl;

cout << "Valor de p (endereço de x): " << p << endl;

cout << "Valor apontado por p: " << \*p << endl;

return 0;

}

**Explicação**

1. **Declaração de Variáveis**: int x = 10; declara uma variável inteira x e inicializa com 10.
2. **Operador de Endereço**: int \*p = &x; declara um ponteiro p que armazena o endereço de x.
3. **Saída**:
   * cout << "Valor de x: " << x << endl; exibe o valor de x.
   * cout << "Endereço de x: " << &x << endl; exibe o endereço de x.
   * cout << "Valor de p (endereço de x): " << p << endl; exibe o valor de p (que é o endereço de x).
   * cout << "Valor apontado por p: " << \*p << endl; exibe o valor apontado por p (que é o valor de x).

**Função strlen()**

A função strlen() é usada para calcular o comprimento de uma string em C++. Ela está definida na biblioteca <cstring>.

**Sintaxe**

size\_t strlen(const char \*str);

* str: ponteiro para a string cuja extensão (comprimento) deve ser calculada.
* Retorna o número de caracteres na string str, excluindo o caractere nulo ('\0').

**Exemplo de Uso**

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char str[] = "Olá, Mundo!";

size\_t comprimento = strlen(str);

cout << "String: " << str << endl;

cout << "Comprimento da string: " << comprimento << endl;

return 0;

}

**Explicação**

1. **Declaração da String**: char str[] = "Olá, Mundo!"; declara e inicializa uma string.
2. **Calculando o Comprimento**: size\_t comprimento = strlen(str); calcula o comprimento da string.
3. **Saída**:
   * cout << "String: " << str << endl; exibe a string.
   * cout << "Comprimento da string: " << comprimento << endl; exibe o comprimento da string.

**Exemplo Completo Combinando Ambos**

Vamos criar um exemplo que usa tanto o operador de endereço (&) quanto a função strlen().

#include <iostream>

#include <cstring>

using namespace std;

int main() {

// Declaração de variáveis

char str[] = "Programação em C++";

int numero = 42;

// Calculando o comprimento da string

size\_t comprimento = strlen(str);

// Usando o operador de endereço

int \*ponteiroNumero = &numero;

// Saída dos resultados

cout << "String: " << str << endl;

cout << "Comprimento da string: " << comprimento << endl;

cout << "Valor de numero: " << numero << endl;

cout << "Endereço de numero: " << &numero << endl;

cout << "Valor do ponteiro (endereço de numero): " << ponteiroNumero << endl;

cout << "Valor apontado pelo ponteiro: " << \*ponteiroNumero << endl;

return 0;

}

**Explicação do Exemplo Completo**

1. **Declaração de Variáveis**:
   * char str[] = "Programação em C++"; declara uma string.
   * int numero = 42; declara um inteiro.
2. **Calculando o Comprimento da String**: size\_t comprimento = strlen(str); calcula o comprimento da string str.
3. **Usando o Operador de Endereço**: int \*ponteiroNumero = &numero; declara um ponteiro ponteiroNumero que armazena o endereço de numero.
4. **Saída**:
   * cout << "String: " << str << endl; exibe a string.
   * cout << "Comprimento da string: " << comprimento << endl; exibe o comprimento da string.
   * cout << "Valor de numero: " << numero << endl; exibe o valor de numero.
   * cout << "Endereço de numero: " << &numero << endl; exibe o endereço de numero.
   * cout << "Valor do ponteiro (endereço de numero): " << ponteiroNumero << endl; exibe o valor do ponteiro (endereço de numero).
   * cout << "Valor apontado pelo ponteiro: " << \*ponteiroNumero << endl; exibe o valor apontado pelo ponteiro (valor de numero).

Este exemplo ilustra como usar o operador de endereço (&) e a função strlen() em C++ para manipular endereços de memória e calcular o comprimento de strings.

Diretiva #define

A diretiva #define é usada para definir macros em C++. Uma macro é uma substituição de texto que é realizada pelo pré-processador antes da compilação do programa.

**Exemplos de #define**

1. **Definindo Constantes:**

#include <iostream>

using namespace std;

#define PI 3.14159

#define DIAS\_NO\_ANO 365

int main() {

cout << "Valor de PI: " << PI << endl;

cout << "Dias no ano: " << DIAS\_NO\_ANO << endl;

return 0;

}

Definindo Macros com Argumentos:

#include <iostream>

using namespace std;

#define AREA\_DO\_CIRCULO(r) (PI \* (r) \* (r))

int main() {

const double PI = 3.14159;

double raio = 5.0;

cout << "Área do círculo com raio " << raio << " é: " << AREA\_DO\_CIRCULO(raio) << endl;

return 0;

}

**Funções**

Funções em C++ são blocos de código reutilizáveis que realizam uma tarefa específica. Elas ajudam a organizar o código e a evitar a repetição.

**Sintaxe Básica de Funções**

#include <iostream>

using namespace std;

// Declaração da função

int soma(int a, int b) {

return a + b;

}

int main() {

int resultado = soma(5, 3);

cout << "Resultado da soma: " << resultado << endl;

return 0;

}

**Funções de Entrada e Saída de Caracteres**

**getche() e getch()**

As funções getche() e getch() são usadas para ler um caractere do teclado. Elas são definidas na biblioteca <conio.h>, que é específica do DOS/Windows e não está disponível em todos os sistemas.

* **getche()**: Lê um caractere do teclado e o exibe no console.
* **getch()**: Lê um caractere do teclado, mas não o exibe no console.

**Exemplo com getche() e getch()**

#include <iostream>

#include <conio.h> // Necessário para getche() e getch()

using namespace std;

int main() {

char c;

cout << "Digite um caractere (getche): ";

c = getche(); // Lê e exibe o caractere

cout << "\nVocê digitou: " << c << endl;

cout << "Digite outro caractere (getch): ";

c = getch(); // Lê o caractere sem exibir

cout << "\nVocê digitou: " << c << endl;

return 0;

}

**getchar() e putchar()**

Estas funções são padrão C e estão disponíveis na biblioteca <cstdio> (ou <stdio.h>).

* **getchar()**: Lê um caractere do teclado.
* **putchar()**: Escreve um caractere no console.

**Exemplo com getchar() e putchar()**

#include <iostream>

#include <cstdio> // Necessário para getchar() e putchar()

using namespace std;

int main() {

char c;

cout << "Digite um caractere: ";

c = getchar(); // Lê um caractere

cout << "Você digitou: ";

putchar(c); // Exibe o caractere

cout << endl;

return 0;

}

**Exemplo Completo**

Vamos criar um exemplo completo que usa #define, funções, e as funções de entrada/saída de caracteres.

#include <iostream>

#include <cstdio>

#include <conio.h>

using namespace std;

#define PI 3.14159

#define AREA\_DO\_CIRCULO(r) (PI \* (r) \* (r))

// Função para calcular a soma de dois inteiros

int soma(int a, int b) {

return a + b;

}

int main() {

// Uso da diretiva #define

double raio = 5.0;

cout << "Área do círculo com raio " << raio << " é: " << AREA\_DO\_CIRCULO(raio) << endl;

// Uso da função soma

int resultado = soma(7, 3);

cout << "Resultado da soma: " << resultado << endl;

// Uso de getchar() e putchar()

char c;

cout << "Digite um caractere: ";

c = getchar();

cout << "Você digitou: ";

putchar(c);

cout << endl;

// Uso de getche() e getch()

cout << "Digite um caractere (getche): ";

c = getche();

cout << "\nVocê digitou: " << c << endl;

cout << "Digite outro caractere (getch): ";

c = getch();

cout << "\nVocê digitou: " << c << endl;

return 0;

}

**Explicação do Exemplo Completo**

1. **Diretiva #define**:
   * Definimos PI e a macro AREA\_DO\_CIRCULO(r) para calcular a área de um círculo.
2. **Função soma**:
   * Definimos a função soma que retorna a soma de dois inteiros.
3. **Entrada/Saída de Caracteres**:
   * Usamos getchar() para ler um caractere e putchar() para exibi-lo.
   * Usamos getche() para ler e exibir um caractere.
   * Usamos getch() para ler um caractere sem exibi-lo.

Este exemplo ilustra como usar #define para definir macros, criar e usar funções, e trabalhar com funções de entrada/saída de caracteres em C++.