Algoritmos e Lógica de Programação

PROF. NILTON

Aula de hoje

Introdução a linguagem C++

- Operadores;
- Bibliotecas;
- Tipos de dados;
- Escopo de um programa;
- Estruturas condicionais;
- Estruturas de repetição;

Introdução

A linguagem C foi desenvolvida a partir do B por Dennis M. Ritchie, do Bell Laboratories, e implementada originalmente em um computador DEC PDP-11, em 1972. De início o C se tornou amplamente conhecido como a linguagem de desenvolvimento do sistema operacional UNIX. Hoje em dia, praticamente todos os grandes sistemas operacionais estão escritos em C e/ou C++ .

Introdução

Os programas em C consistem em módulos ou elementos chamados funções. Você pode programar todas as funções de que precisa para formar um programa C, mas a maioria dos programadores C tira proveito de um excelente conjunto de funções chamado C Standard Library (Biblioteca Padrão do C).

Dessa forma, há na realidade duas partes a serem aprendidas no "mundo" do C. A primeira é a linguagem C em si, e a segunda é como usar as funções do C Standard Library.

Introdução

A linguagem C++ surgiu em meados dos anos 80, a partir de uma evolução da linguagem C.

A linguagem C++ tem como característica o multiparadigma, ou seja, pode ser trabalhada tanto como linguagem estruturada ou orientada a objetos, o que não era possível em C.

Streams e Entrada/Saída

<iostream>

Este cabeçalho é responsável pela manipulação de fluxo de dados padrão do sistema (entrada padrão, saída padrão e saída de erros padrão).

Ele representa uma evolução do cabeçalho <stdio.h> da linguagem C. Permite o uso de operadores de deslocamento de bis (<< e >>) juntamente com os objetos (cin, cout, cerr, e clog) para envio e recebimento de dados dos fluxos de entrada, saída, erro com e sem buffer.

Operadores aritméticos

OPERAÇÃO	EM C/C++
Soma	+
Subtração	-
Multiplicação	*
Divisão	/
Resto da Divisão	%

Operadores aritméticos

OPERAÇÃO	EM C/C++
Igual	==
Diferente	!=
Menor	<
Menor ou igual a	<=
Maior	>

Tipos de dados

Números:

- Inteiros curtos → short
- Inteiros normais → int
- Inteiros longos → long
- Reais simples → float
- Reais longos → Double

Tipos de dados

Letras (literais):

- Um caractere → char
- Conjunto de caracteres → string

Lógicos:

- Booleanos → boolean

Tipos de Dados

Tipo de Dado	Significado	Tamanho (em bytes)	Intervalo de valores
char	Caractere	1	-128 a 127
short int	Inteiro curto	2	-32.768 a 32.767
int	Inteiro	2 (16 bits) 4 (32 bits)	-32.768 a 32.767 -2.147.483.648 a 2.147.483.647
long int	Inteiro longo	4	-2.147.483.648 a 2.147.483.647
float	Flutuante (real)	4	$3.4*10^{-38}$ à $3.4*10^{38}$
double	Flutuante duplo	8	$1.7*10^{-308}$ à $1.7*10^{308}$
long double	Flutuante duplo longo	10	$3.4*10^{-4932}$ à $3.4*10^{4932}$

Caracteres de controle

Controle	Efeito
\a	(alarm) Soa o alarme do terminal
\b	(back space) muda o cursor para a coluna anterior
\n	(new line) muda o cursor para a próxima linha
\r	(return) muda o cursor para o inicio da linha
\t	(tab) muda o cursor para a próxima marca de tabulação
\’	Exibe um apostrofo
\”	Exibe uma aspa
\\	Exibe uma barra invertida

Escopo de um programa em C/C++

Um **escopo** é uma representação básica. Neste caso iremos expressar os pontos mais básicos de um código fonte em c++.

#include <iostream> //declaração de bibliotecas

using namespace std; // utilização do namespace (espaço de nome) para organizar as classes

int main() // função principal

{ // inicio do bloco de instruções da função principal

// **intruções** → geralmente declaramos todas as variáveis antes de elaborar as demais instruções.

} // fim do bloco de instruções da função principal

Declaração de variáveis

//declaração de variáveis

tipo nome_da_variavel;

Exemplos:

int numero;

string nome;

char sexo;

float media;

Comando de saída

Objeto "**cout**'"

O objeto **cout** representa o stream de saída no C++. Este stream é uma espécie de sequência (fluxo) de dados a serem impressos na tela.

Para realizar a impressão, usa-se o "operador de inserção" que "insere" dados dentro do stream.

```
cout << "Imprimindo o famoso HELLO WORLD!!!\n";
```

Comando de entrada

Objeto "**cin**"

O objeto **cin** representa o stream de entrada no C++. Ele realiza a leitura de uma sequência de dados, sem espaços e sem tabulações, vindas do teclado.

Para coletar estes dados armazenados, usa-se o "operador de extração" que "extrai" dados do stream.

```
cout << "Lendo o primeiro número...: ";
cin >> Num1;
```

Ao lado temos um algoritmo escrito em C++, que tem como função mostrar uma mensagem na tela.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
  string nome;
  int idade;
  cout << “Digite seu nome:  “;
  cin >> nome;
  cout << “Digite sua idade:  ”;
  cin >> idade;
  cout << “Então você já possui “<< idade << ” anos.”;
  system(“pause”);
}
```

Console

system()

Utilizando o comando system(), podemos executar qualquer comando que pode ser executado no terminal do Sistema Operacional.

system(“pause”) → faz uma pausa na execução do programa

system(“cls”) → limpa a tela (console)

Estrutura condicional simples

Essa estrutura serve para selecionar e executar um ou mais comandos a partir de uma condição. Essa condição é avaliada e se for verdadeira o comando é executado.

```
if (condição)
    comando 1;
```

Estrutura condicional composta

Essa estrutura serve para selecionar e executar apenas um entre dois comandos/blocos alternativos.

Para isso a condição é avaliada e:

se for verdadeira, ***então*** apenas o primeiro comando/bloco é executado

senão, apenas o segundo comando /bloco é executado.

if (condição)

comando 1;

else

comando 2;

Estrutura condicional encadeada

Essa estrutura serve para selecionar e executar apenas um entre varios comandos/blocos alternativos.

if (condição)

comando 1;

else if (condição2)

comando 2;

else if (condiçãao3)

comando 3;

Uso de blocos

Em C/C++, quando temos mais de um comando a ser executado para uma determinada condição, devemos agrupar esses comandos em um bloco delimitado por chaves { }.

```
if ( condição ) {
    comando 1;
     ...
    comando n;
}
else
    comando 2;
```

```
if ( condição )
    comando 1;
else {
    comando 2;
    ...
    comando n;
}
```

```
if ( condição ) {
    comando 1;
     ...
    comando n;
}
else {
    comando 2;
    ...
    comando n;
}
```

Seleção múltipla – Escolha/Caso

Uma estrutura de seleção múltipla é composta de varias estruturas de seleção simples encadeadas com as seguintes propriedades:

- Todas as condições nas decisões são de igualdade;
- Todas as condições comparam uma mesma *expressão* a uma *constante*;
- Todas as constantes consideradas são do tipo inteiro ou caracter.

Sintaxe:

```
switch ( expressão ) {
    case constante1: comando1; break;
    case constante2: comando2; break;
    ...
    default: comando_n: break
}
```

Estruturas de Repetição

Acumuladores e Contadores

Operação	Pseudocodigo	Programa em C
Soma	a ← a + expr	a += expr
Subtração	a ← a - expr	a -= expr
Multiplicação	a ← a * expr	a *= expr
Divisão	A ← a / expr	a /= expr

Operação	Pseudocodigo	Programa em C
Incremento	c ← c + 1	c++
Decremento	c ← c + 1	c--

Estrutura de repetição contada

A estrutura de repetição contada, serve para repetir a execução de um comando por um determinado numero de vezes. Para saber quando o total de repetições já foi atingido, a estrutura de repetição contada usa um ***contador***. Nesta estrutura:

- Primeiramente o contador é iniciado com um valor específico.
- Depois o contador é testado: se o total de repetições ainda não foi atingido, a repetição continua; caso contrário, ela termina.
- A cada nova repetição o contador é alterado.

```
for ( inicia; testa; altera )
   comando;
```

```
for ( inicia; testa; altera){
   comando1;
   ...
   comandoN;
}
```

```
for(int i = 0; i < 10; i++) {
   comando1;
   ...
   comandoN
}
```

Estrutura de repetição Pré-Testada

Esta estrutura serve para executar um comando, repetidamente, enquanto uma determinada condição for verdadeira. Devemos observar que como a condição é testada antes do comando ser executado, se ela for inicialmente falsa, o comando dentro da repetição jamais é executado.

```
while ( condição ) {
   comando1;
   ...
   comandoN;
}
```

```
int n=1;

while ( n > 0 ) {
   r = n % 10;
   n = n / 10;
   cout << r;
}
```

Estrutura de repetição Pós-Testada

A estrutura pós-testada, serve para executar um comando, repetidamente, até que uma determinada condição se torne falsa. Essa estrutura é usada quando não sabemos de antemão quantas vezes o comando deve ser repetido, mas precisamos que ele seja executado pelo menos uma vez.

Em geral a estrutura pós-testada é usada para:

- Garantir consistência de entrada de dados.
- Repetir um processo com confirmação do usuário.
- Implementar processos orientados a menus.

Estrutura de repetição Pós-Testada

```
do {
   comando1;
   ...
   comandoN;
} while ( condição );
```

```
do {

   cout << “Digite um numero”;
   cin >> n;

} while ( n > 0);
```

Referências

Site

http://www.cplusplus.com/reference

Livro: Treinamento em Linguagem C, 2ª edição - Pearson