C/C++

Fundamentos

Prof. Edson Alves - UnB/FGA

2018

Sumário

Conceitos elementares

C

- C é uma linguagem de programação estaticamente tipada, de forma livre, imperativa e de propósito geral
- Foi criada por Dennis Ritchie em 1972 no laboratório da Bell para desenvolvimento em sistemas operacionais Unix
- **Leitura complementar**: RITCHIE, Dennis M. *The Development of the C Language*. AT&T Bell Laboratories. Murray Hill, New Jersey. 1993.

O arquivo source.c pode ser compilado com o GCC usando a seguinte linha de comando:

```
$ gcc -o prog -W -Wall -Wextra -pedantic -O2 source.c
```

C++

- C++ é uma linguagem de programação estaticamente tipada, de forma livre, multi-paradigma e de propósito geral
- Foi criada por Bjarne Stroustrup em 1979 no laboratório da Bell, inicialmente como uma extensão da linguagem C
- **Leitura complementar**: STROUSTRUP, Bjarne. *A History of C++: 1979-1991*. AT&T Bell Laboratories. Murray Hill, New Jersey. 1994.

O arquivo source.cpp pode ser compilado com o GCC usando a seguinte linha de comando:

```
$ g++ -o prog -W -Wall -Wextra -std=c++17 -O2 source.cpp
```

Hello World!

- A prática recorrente é apresentar uma linguagem através do programa mais elementar de todos: o *Hello World!*
- O propósito deste programa é ilustrar o mecanismo de saída (*output*) da linguagem
- Uma das maneiras de se medir (de forma superficial) a complexidade (sintática) de uma linguagem é contabilizar o número de linhas do *Hello World!*
- *Hello World!* em Python 3:

```
1 print("Hello World!")
```

Hello World! em C/C++

```c
#include <stdio.h>

int main()
{
    printf("Hello World!\n");

    return 0;
}
```

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello World!" << std::endl;

    return 0;
}
```

Variáveis

Sintaxe para declaração de variaveis

```
[armazenamento|acesso|modificador]tipo nome[=valor inicial];
```

- Variáveis são valores que podem ser modificados ao longo da execução do programa
- Em C, a cada variável é associado um dos 5 tipos primitivos de dados: **`char`**, **`short`**, **`int`**, **`double`**, **`float`**
- Palavras-chave associadas ao:
  - armazenamento: **`extern`**, **`static`**, **`register`**, **`auto`**
  - acesso: **`const`**, **`volatile`**
  - modificador: **`signed`**, **`unsigned`**, **`long`**, **`short`**

Classificação das variáveis

Aspecto	Classificação	Descrição
Localização	globais	Podem ser acessadas em qualquer função
	locais	Podem ser acessadas apenas no bloco em que foram declaradas
	parâmetros de função	Podem ser acessados apenas pela função. São preenchidos na chamada da função
Acesso	gerais	Leitura e escrita
	constantes	Apenas leitura. Devem ser declaradas com um valor inicial
	voláteis	Podem ser modificadas por programas externos
Armazenamento	externas	A definição ocorre em outro arquivo
	estáticas	O armazenamento é alocado uma única vez
	registradores	O armazenamento é feito em registradores, não em memória

Exemplo de variáveis

```c
#include <stdio.h>

float media = 0.0f;                                 /* Variável global */

void atualiza_media(float nota) {                   /* Parâmetro de função */
    static int quantidade_notas = 0;                /* Variável estática */
    media = media*quantidade_notas + nota;
    quantidade_notas++;
    media /= quantidade_notas;
}

int main() {
    float nota1 = 3.8f, nota2 = 4.2f;               /* Variáveis locais */

    atualiza_media(nota1);
    atualiza_media(nota2);

    printf("A media e igual a %3.2f\n", media);

    return 0;
}
```

Estruturas de seleção e controle

if/else

Sintaxe para declaração do `if`

```
if (condicao) {
    bloco if
} else {
    bloco else
}
```

- Se a condição for verdadeira, o bloco **`if`** será executado
- Caso contrário, o bloco **`else`** será executado, se existir
- Operador ternário: a variável assume um dos dois valores listados, a depender do valor lógico da condição: `A`, se verdadeira; `B`, se falsa.
- Sintaxe: `[variável = ] condição ? A : B;`
- Ambos valores `A` e `B` devem ter o mesmo tipo

Exemplo de uso de if/else

```cpp
#include <iostream>

int sinal(long int numero) {
    if (numero >= 0)
        return 1;
    else
        return -1;
}

int main() {
    long numero;

    std::cout << "Insira um número: ";
    std::cin >> numero;

    std::cout << "Sinal do numero: " << (sinal(numero) > 0 ? '+' : '-')
        << std::endl;

    return 0;
}
```

switch

Sintaxe para declaração do switch

```
switch (valor) {
case valor1:
    [comandos1; ...]
    break;

case valor2:
    [comandos2; ...]
    break;

...

case valorN:
    [comandosN; ...]
    break;

[default:]
    [comandos; ...]
}
```

switch

- Se `valor` for igual a um dos valores `valor1, ..., valorN` descritos, serão executados os comandos que se seguem a cláusula **`case`** associada até que se encontre um comando **`break`** ou termine o bloco do **`switch`**
- A cláusula **`default`**, que é escolhida caso valor não corresponda a nenhum valor listado, é opcional
- O `valor` indicado deve ser do tipo inteiro (**`char`** ou **`int`**)

Exemplo de uso de switch

```c
#include <stdio.h>

typedef enum {II, SR, MI, MM, MS, SS} Mencao;

Mencao calcula_mencao(float nota1, float nota2)
{
    float media = (nota1 + nota2)/2.0f;

    if (!media)
        return SR;
    else if (media < 2)
        return II;
    else if (media < 5)
        return MI;
    else if (media < 7)
        return MM;
    else if (media < 9)
        return MS;
    else
        return SS;
}
```

Exemplo de uso de switch

```
23 int main()
24 {
25     float nota1, nota2;
26     Mencao mencao;
27 
28     printf("Insira duas notas: ");
29     scanf("%f %f", &nota1, &nota2);
30 
31     switch (calcula_mencao(nota1, nota2)) {
32     case II:
33     case SR:
34     case MI:
35         printf("Aluno reprovado\n");
36         break;
37 
38     default:
39         printf("Aluno aprovado\n");
40     }
41 
42     return 0;
43 }
```

for

Sintaxe para declaração do for

```
for ([inicializacao]; [condicao]; [incremento])
{
    [ bloco de comandos ];
}
```

- O laço **for** é começa com a etapa de `inicialização`
- Em seguida, é verificada a condição: se falsa, o laço é encerrado; caso contrário, é executado o `bloco de comandos`
- Finalizada a execução do `bloco de comandos`, é executado o `incremento` e a condição é novamente testada, e assim sucessivamente até o encerramento do laço

for

- A `inicialização`, a `condição` e o `incremento` são opcionais
- O laço **`for`**, assim como os demais laços, pode ser interrompido a qualquer momento através de um comando **`break`**
- O comando **`continue`** força o encerramento prematuro do bloco de comandos, levando a execução imediatamente para a avaliação do `incremento`
- Sem o uso de um **`break`**, o laço poderá se executado infinitamente (o que resultará no travamento do console ou numa falha de segmentação) caso a condição não se torne verdadeira em algum ponto do laço

Exemplo de uso do for

```cpp
#include <iostream>

bool is_prime(long N) {
    if (N == 2) return true;
    if (N < 2 or not (N & 1)) return false;

    for (long d = 3; d * d <= N; d += 2)
        if (not (N % d))
            return false;

    return true;
}

int main() {
    long N = 1013;

    std::cout << N << (is_prime(N) ? " e " : " nao e ")
        << "um numero primo" << std::endl;

    return 0;
}
```

while

Sintaxe para declaração do while

```
while (condicao)
{
    [ bloco de comandos ];
}
```

- O laço **while** começa com a verificação da condição: se falsa, o laço é encerrado; caso contrário, é executado o bloco de comandos
- Finalizada a execução do bloco de comandos, a condição é novamente testada, e assim sucessivamente, até o encerramento do laço
- O laço **while**, assim como os demais laços, pode ser interrompido a qualquer momento através de um comando **break**
- O comando **continue** força o encerramento prematuro do bloco de comandos, levando a execução imediatamente para a avaliação da **condição**

Exemplo de uso de while

```c
#include <stdio.h>

long nextFibonacci() {
    static long prev = 0;
    static long next = 1;

    long temp = prev + next;
    prev = next;
    next = temp;

    return prev;
}

int main() {
    long N = 1000000, next;

    while ((next = nextFibonacci()) < N)
        printf("%ld\n", next);

    return 0;
}
```

Funções

Funções

Sintaxe para declaração de uma função

```
tipo_retorno nome_funcao([lista_de_parametros])
{
    [ implementacao da funcao ];
}
```

- Uma função constitui o bloco básico de um programa estruturado. Nas funções ocorrem todo o processamento do programa
- O tipo do retorno especifica qual é o tipo da variável que será retornada pela função
- A lista de parâmetros é opcional, e pode conter N elementos, separados por vírgula, onde cada elemento é descrito pelo tipo da variável, seguido de seu nome
- O corpo da função pode conter expressões, estruturas de controle e chamadas à outras funções.

Exemplo de uso de funções

```c
#include <stdio.h>

double seno(double angle) {
    double value = angle, term = angle;
    int signal = -1, i = 0;

    for (i = 3; i < 30; i += 2) {
        term = term/((i - 1)*i);
        term *= angle*angle;
        value += signal*term;
        signal *= -1;
    }

    return value;
}

int main() {
    printf("O seno de 1.75 é igual a %.15f\n", seno(1.75));

    return 0;
}
```

Referências

1. **KERNIGHAN**, Bryan; **RITCHIE**, Dennis. *The C Programming Language*, 1978.
2. **RITCHIE**, Dennis. *The Development of the C Language*. AT&T Bell Laboratories. Murray Hill, New Jersey. 1993.
3. **STROUSTROUP**, Bjarne. *A History of C++: 1979-1991*. AT&T Bell Laboratories. Murray Hill, New Jersey. 1994.
4. **STROUSTROUP**, Bjarne. *The C++ Programming Language*, 2013.
5. C++ Reference[1].

[1] https://en.cppreference.com/w/