Programação Estruturada

Introdução à linguagem C

Professores Emílio Francesquini e Carla Negri Lintzmayer

2018.Q3

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

UFABC

Programação estruturada

O que é programação estruturada

Técnica/paradigma de programação que visa o desenvolvimento de algoritmos por meio de estruturas de controle visando clareza e qualidade

- Estrutura sequencial
- Estrutura condicional
- Estrutura de repetição
- Modularização

Exemplos de linguagens que suportam o paradigma estruturado: Pascal, Ada, Java, C, C++.

A linguagem C

Sobre a linguagem C

- Criada em 1972 por Dennis Ritchie
  - The C Programming Language, Kernighan e Ritchie
- Linguagem compilada, de propósito geral, estruturada, imperativa, procedural
- Várias especificações (**ANSI**, C89, C99, C11)
  - Neste curso nos concentraremos na especificação ANSI.

Programas em C

- Um programa em C é conjunto de um ou mais arquivos textos, contendo declarações e operações da linguagem (código fonte)
- Consiste em uma ou mais *funções*
- A única obrigatória é denominada **main()**

Programas em C

```c
#include <stdio.h>

int main() {
    printf("Hello world!\n");
    return 0;
}
```

Variáveis

Variáveis

Definição

Variáveis são locais onde armazenamos valores.
Toda variável é caracterizada por um **nome**, que a identifica em um programa, e por um **tipo**, que determina o que pode ser armazenado naquela variável.

- Durante a execução do programa, um pedacinho da memória corresponde às variáveis
- Em C, existem 5 tipos básicos de variáveis: `char`, `int`, `float`, `double` e `void`[1].

[1] Estritamente falando, `void` não é um tipo, mas sim uma palavra reservada utilizada para marcar a ausência dos outros tipos

Declarando uma variável em C

Comando de declaração

```
tipo_variável nome_variável;
```

Exemplos corretos:

- `int soma;`
- `float preco_abacaxi;`
- `char resposta;`

Exemplos incorretos (por quê?):

- `soma int;`
- `float preco_abacaxi`

Variáveis inteiras

Variáveis inteiras

Utilizadas para armazenar valores inteiros, como 13 ou 1102 ou 24.

Abaixo temos os **tipos da linguagem C** que servem para armazenar inteiros:

int Inteiro cujo comprimento depende do processador. É o mais utilizado. Em processadores de 32 bits (4 bytes) pode armazenar valores entre -2.147.483.648 e 2.147.483.647.

unsigned int Inteiro cujo comprimento depende do processador e que armazena somente valores positivos. Em processadores de 32 bits pode armazenar valores entre 0 e 4.294.967.295.

Pergunta: Como fica o caso dos processadores de 64 bits?

Variáveis inteiras

long int Inteiro que ocupa 64 bits (8 bytes) em computadores Intel de 64 bits, e pode armazenar valores entre aprox. -9×10^{18} e aprox. 9×10^{18}

unsigned long int Inteiro que ocupa 64 bits em computadores Intel de 64 bits, e armazena valores entre 0 e aprox. 18×10^{18}

short int Inteiro que ocupa 16 bits (2 bytes) e pode armazenar valores entre -32.768 e 32.767

unsigned short int Inteiro que ocupa 16 bits e pode armazenar valores entre 0 e 65.535

Variáveis inteiras

Exemplos de declaração de variáveis inteiras:

- `int numVoltas;`
- `int ano;`
- `unsigned int quantidadeChapeus;`

Exemplos inválidos:

- `int int numVoltas;`
- `unsgned int ano;`

Declarando várias variáveis

Você pode declarar várias variáveis de um mesmo tipo em uma linha.

Basta separar as variáveis por vírgula:

- `int numVoltas, ano;`
- `unsigned int val1, val2, val3, val4;`

Variáveis do tipo caractere

Variáveis caracteres

Utilizadas para armazenar uma letra ou outro símbolo existente em textos. Ex: `'a'`, `'3'`, `','`, `'+'`.

Exemplos de declaração:

- `char umaLetra;`
- `char S_ou_N;`

Variáveis de tipo ponto flutuante

Variáveis de ponto flutuante

Armazenam valores reais, mas possuem problemas de precisão pois há uma quantidade limitada de memória para armazenar um número real. Ex: 2.1345 ou 9098.123.

Abaixo temos os **tipos da linguagem C** que servem para armazenar números de ponto flutuante:

float Utiliza 32 bits, e na prática tem precisão de aproximadamente 6 casas decimais (depois do ponto). Pode armazenar valores entre 3.4×10^{-38} e 3.4×10^{38}

double Utiliza 64 bits, e na prática tem precisão de aproximadamente 15 casas decimais. Pode armazenar valores entre 1.7×10^{-308} e 1.7×10^{308}

Variáveis de tipo ponto flutuante

Exemplos de declaração:

- `float salario, resultado;`
- `double cotacaoDolar;`

Regras para nome de variável em C

- **Deve** começar com uma letra (maiúscula ou minúscula) ou subscrito (_)
- **Nunca** pode começar com um número
- Pode conter letras maiúsculas, minúsculas, números e subscrito
- Não pode conter os símbolos `{ ( + - * / \ ; . , ?`
- Letras maiúsculas e minúsculas são diferentes:

```
1 int c, C;
```

Regras para nome de variável em C

As seguintes palavras já tem um significado na linguagem C (são reservadas) e por esse motivo não podem ser utilizadas como nome de variáveis:

`auto`	`double`	`int`	`struct`
`break`	`else`	`long`	`switch`
`case`	`enum`	`register`	`typedef`
`char`	`extern`	`return`	`union`
`const`	`float`	`short`	`unsigned`
`continue`	`for`	`signed`	`void`
`default`	`goto`	`sizeof`	`volatile`
`do`	`if`	`static`	`while`

Atribuição

Atribuindo valores a variáveis

Comando de atribuição

`nome_variável = valor;`

ou

`nome_variável = expressão;`

Exemplos:

```
1 int a;
2 float c;
3 a = 5;
4 c = 67.89505456;
5 a = 3 + 5;
```

Atribuir um valor de uma expressão para uma variável significa calcular o valor daquela expressão e então copiar aquele valor para a variável (nessa ordem).

Atribuição

- O sinal de igual (=) no comando de atribuição é chamado de **operador de atribuição**
- Veremos outros operadores mais adiante

À esquerda do operador de atribuição deve existir somente o nome de uma **variável**

=

À direita, deve haver uma **expressão** cujo valor será calculado e armazenado na variável

Variáveis e constantes

Constantes

São valores previamente determinados e que, por algum motivo, devem aparecer dentro de um programa

- Constantes também possuem um tipo
- Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo `string`, que corresponde a uma sequência de caracteres
- Exemplos de constantes:

```
1  85, 0.10, 'c', "Hello, world!"
```

Expressões simples

Uma constante é uma expressão e, como tal, pode ser atribuída a uma variável (ou ser usada em qualquer outro lugar onde uma expressão seja válida)

Exemplos:

```
1 int a;
2 a = 10;
3 char b;
4 b = 'F';
5 double c;
6 c = 3.141592;
```

Expressões simples

Uma variável também é uma expressão e pode ser atribuída a outra variável

Exemplo:

```
1 int a, b;
2 a = 5;
3 b = a;
```

Expressões simples

```
1  int a, b, c;
2
3  a = 5 + 5 + 10;
4  b = 6;
5  c = a + b;
6  a = c + 4;
```

Qual é o valor de `a`, `b` e `c`?

Exemplos de atribuição

A declaração de uma variável **sempre** deve ocorrer antes de seu uso

```
1  int a, b;
2  float f;
3  char h;
4
5  a = 10;
6  b = -15;
7  f = 10.0;
8  h = 'A';
9
10 a = b;
11 f = a;
12 a = (b + a);
```


Qual o valor final na variável a?

Exemplos errados de atribuição

```
1  int a, b;
2  float f, g;
3  char h;
4
5  a b = 10;
6  b = -15
7  d = 90;
```

Quais são os erros?

Estrutura de um programa em C

Estrutura básica de um programa em C

```
1  Declaracao de bibliotecas usadas
2
3  Declaracao de variaveis
4
5  int main() {
6      Declaracao de variaveis
7
8      Comandos
9
10     return 0;
11 }
```

Estrutura básica de um programa em C

```c
#include <stdio.h>

int main() {
    int a;
    int b, c;

    a = 7 + 9;
    b = a + 10;
    c = b - a;

    return 0;
}
```

Exercícios

Exercício 1

Qual o valor armazenado na variável **a** no fim do programa?

```c
int main() {
    int a, b, c, d;

    d = 3;
    c = 2;
    b = 4;
    d = c + b;
    a = d + 1;
    a = a + 1;

    return 0;
}
```

Exercício 2

Qual erro existe neste programa?

```
1  int main () {
2      int a, b;
3      double c, d;
4      int g;
5
6      d = 3.0;
7      c = 2.4142;
8      b = 4;
9      a = 5 * b;
10     d = b + 90;
11     e = c * d;
12     a = a + 1;
13
14     return 0;
15 }
```

Expressões e operadores aritméticos

Expressões

- Vimos que constantes e variáveis são expressões
- Uma expressão também pode ser um conjunto de operações aritméticas, lógicas ou relacionais utilizadas para fazer “cálculos” sobre os valores das variáveis.

Exemplo de expressão:

```
1 a + b
```

Calcula a soma de **a** e **b**.

Expressões aritméticas

- Os operadores aritméticos são: `+`, `-`, `*`, `/`, `%`
- *expressão* `+` *expressão* : Calcula a soma de duas expressões.
  Ex: `10 + 15;`
- *expressão* `-` *expressão* : Calcula a subtração de duas expressões.
  Ex: `5 - 7;`
- *expressão* `*` *expressão* : Calcula o produto de duas expressões.
  Ex: `3 * 4;`

Expressões aritméticas

- *expressão* / *expressão* : Calcula a divisão de duas expressões.
  Ex: `4 / 2;`
- *expressão* % *expressão* : Calcula o resto da divisão (inteira) de duas expressões.
  Ex: `5 % 2;`
- - *expressão* : Inverte o sinal da expressão.
  Ex: `-5;`

Expressões aritméticas

Mais sobre o operador resto da divisão %:

- Quando computamos "a dividido por b", isto tem como resultado um valor p e um resto $r < b$ que são únicos tais que

$$a = p * b + r$$

- Ou seja, a pode ser dividido em p partes inteiras de tamanho b, e sobrará um resto $r < b$.

Exemplos:

5%2 tem como resultado o valor 1.
15%3 tem como resultado o valor 0.
1%5 tem como resultado o valor 1.
19%4 tem como resultado o valor 3.

Expressões aritméticas

No exemplo abaixo, quais valores serão impressos?

```c
#include <stdio.h>

int main() {
    printf("%d\n", 29 % 3);
    printf("%d\n", 4 % 15);

    return 0;
}
```

Expressões aritméticas

Mais sobre o operador de divisão /:

- Quando utilizado sobre valores inteiros, o resultado da operação de divisão será inteiro. Isto significa que a parte fracionária da divisão será desconsiderada.
  - 5/2 tem como resultado o valor 2.
- Quando pelo menos um dos operandos for ponto flutuante, então a divisão será fracionária. Ou seja, o resultado será a divisão exata dos valores.
  - 5.0/2 tem como resultado o valor 2.5.

Expressões aritméticas

No exemplo abaixo, quais valores serão impressos?

```c
#include <stdio.h>

int main() {
    int a = 5, b = 2;
    float c = 5.0, d = 2.0;

    printf("%d\n",a/b);
    printf("%f\n", a/d);
    printf("%f\n", c/d);

    return 0;
}
```

Expressões aritméticas

- As expressões aritméticas (e todas as expressões) operam sobre outras expressões.
- É possível compor expressões complexas como por exemplo:
  ```
  a = b * ((2/c) + (9 + d*8));
  ```
- Qual o valor da expressão `5 + 10 % 3`?
- E da expressão `5 * 10 % 3`?

Precedência

- Precedência é a ordem na qual os operadores serão avaliados quando o programa for executado. Em C, os operadores são avaliados na seguinte ordem:
  - `*` e `/`, na ordem em que aparecerem na expressão.
  - `%`
  - `+` e `-`, na ordem em que aparecerem na expressão.
- Exemplo: `8 + 10*6` é igual a 68.

Alterando a precedência

- (*expressão*) também é uma expressão, que calcula o resultado da expressão dentro dos parênteses primeiro.
  - 5 + 10 % 3 é igual a 6
  - (5 + 10) % 3 é igual a 0
- Você pode usar quantos parênteses desejar dentro de uma expressão.
- Use sempre parênteses em expressões para deixar claro em qual ordem a expressão é avaliada!

Operadores $++$ e $--$

Incremento(++) e decremento(--)

- É muito comum escrevermos expressões para incrementar/decrementar o valor de uma variável por 1.

```
1 a = a + 1;
2 a = a - 1;
```

- Em C, o operador unário ++ é usado para incrementar de 1 o valor de uma variável.

```
1 a++;
```

- O operador unário -- é usado para decrementar de 1 o valor de uma variável.

```
1 a--;
```

Exercícios

Exercício

Crie um programa que computa e imprime a soma, a diferença, a multiplicação e divisão dos dois números do tipo double.

Informações extras

Informações extras: incremento($++$) e decremento($--$)

Há uma diferença quando estes operadores são usados à esquerda ou à direita de uma variável **e** fazem parte de uma expressão maior:

- `++a`: Neste caso o valor de **a** será incrementado antes e só depois o valor de **a** é usado na expressão.
- `a++`: Neste caso o valor de **a** é usado na expressão, e só depois é incrementado.
- A mesma coisa acontece com o operador $--$.

Informações extras: incremento(++) e decremento(--)

O que o programa abaixo imprime?

```c
#include <stdio.h>

int main() {
    int a = 5, b;

    b = ++a;
    printf("b: %d\n", b);
    printf("a: %d\n", a);

    b = a++;
    printf("b: %d\n", b);
    printf("a: %d\n", a);

    return 0;
}
```

Informações extras: atribuições simplificadas

Uma expressão da forma

```
1 a = a + b;
```

Pode ser simplificada como

```
1 a += b;
```

A variável à qual o valor é atribuído faz parte da expressão.

Informações extras: atribuições simplificadas

Comando	Exemplo	Corresponde a:
+=	a += b;	a = a + b;
-=	a -= b;	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;