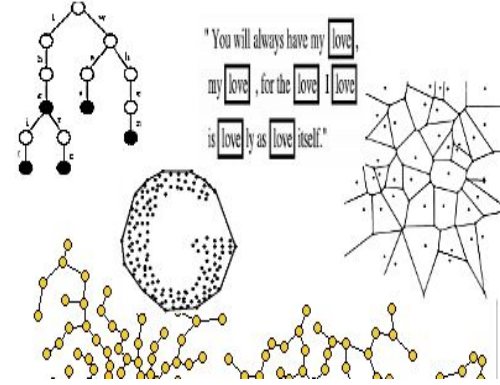


Programação Estruturada

TCC-00347



Prof. Aline Paes / alinepaes@ic.uff.br

Linguagem C - comandos básicos
(Material do livro HeadFirst C - capítulo 1)



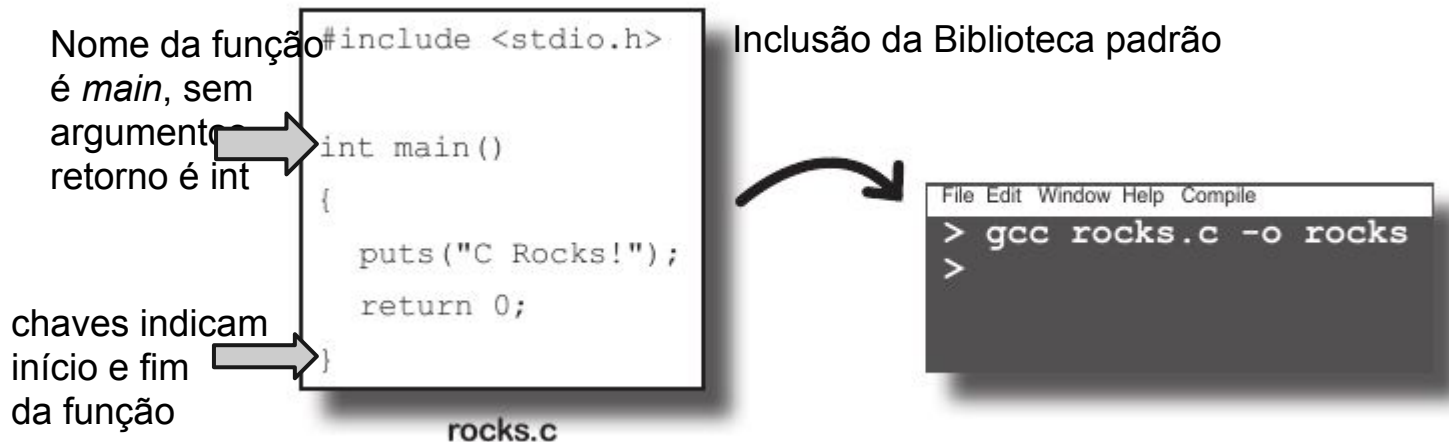
Universidade Federal Fluminense



Tópicos

- Estrutura básica
- Leitura e escrita
- Decisões
 - if
 - if... else if... else
 - switch
- Repetição
 - for
 - while
 - do...while

Um programa em C



OBS: função *main* é o ponto de partida para execução do programa. Todo programa precisa de uma função *main*

Compilação de programa em C

- Compilação X interpretação
 - Compilação de um programa em C
 - > gcc -o prog prog.c
 - > ./prog
 - Mais de um arquivo fonte
 - > gcc -o prog prog.c fun.c
- Ou
- > gcc -c prog.c
 - > gcc -c fun.c
 - > gcc -o prog prog.o fun.o

Declaração de variáveis

- tipo nome;
 - nome = valor;
- ou
- tipo nome = valor;

Tipos básicos

Tipo	Tamanho	Representatividade
char	1 byte	-128 a 127
unsigned char	1 byte	0 a 255
short int	2 bytes	-32768 a 32767
unsigned short int	2 bytes	0 a 65535
long int	4 bytes	-2147483648 a 2147483647
unsigned long int	4 bytes	0 a 4294967295
int	Depende da máquina (32 bits ou 64 bits)	
float	4 bytes	$\pm 10^{-38}$ a 10^{38}
double	8 bytes	$\pm 10^{-308}$ a 10^{308}

* sizeof retorna o número de bytes

O que significa cada linha?

```
/* O que esse programa faz?  
*/  
  
# include <stdio.h>  
  
int main( ) {  
    int card_count = 11;  declara uma variável inteira e associa a ela o valor 11  
    if (card_count > 10)  estrutura condicional  
        puts("Aumente sua aposta!");  Escrita na tela  
    return 0;  
}
```

Operadores aritméticos

<code>+=</code>	soma e atribuição	<code>teeth += 4;</code>	<code>teeth = teeth + 4</code> <code>teeth = 8</code>
<code>-=</code>	subtração e atribuição	<code>teeth -= 2;</code>	<code>teeth = teeth - 2</code> <code>teeth = 6</code>
<code>++</code>	incrementa 1	<code>teeth ++;</code>	<code>teeth = teeth + 1</code> <code>teeth = 7</code>
<code>--</code>	decrementa 1	<code>teeth --;</code>	<code>teeth = teeth - 1</code> <code>teeth = 6</code>
<code>%</code>	Resto da divisão	<code>int x = teeth % 2;</code>	

- Além dos óbvios, com divisão inteira

Operadores aritméticos

- Qual o valor atribuído a x e a y ?
 - `int n = 5;`
 - `int x = n++;` // incrementa depois de usar o valor
 `// x = 5`
 `// n = 6`
 - `int y = ++n;` // incrementa antes de usar o valor
 `// n = 7`
 `// y = 7`

Operadores relacionais

Operador	Semântica
<	menor
>	maior
<=	Menor ou igual
>=	Maior ou igual
==	igual
!=	diferente

* Falso é 0 e qualquer valor diferente de 0 é verdadeiro. Não existe booleano

Leitura e escrita

```
/* Exemplo de leitura do teclado e saída formatada
*/

# include <stdio.h>

int main() {
    int decks;  variável inteira, sem valor inicial
    puts("Digite um valor para decks: ");
    scanf("%i", &decks);  O que o usuário digitar será guardado em decks
    if (decks < 1) {
        puts("Valor inválido!");
        return -1;
    }
    printf("Existem %i cartas\n", (decks * 52));  escrita formatada: um valor inteiro entra no lugar de %i
    return 0;
}
```

Bloco, englobado por { }

Você pode incluir quantos parâmetros quiser no printf, mas lembre-se de obedecer a correspondência entre o caracter especial e o valor a ser impresso

Formatação de caracteres para scanf e printf

- %d - inteiro na base decimal com sinal
- %i - o mesmo, mas se tiver 0x na frente, vai interpretar como hexa e se tiver 0 na frente vai interpretar como octal
- %u - unsigned int
- %f - ponto flutuante
- %o - octal
- %s - string. Inclui /0 no fim
- %c - char
- %h - half (para short); %l - long

O que será impresso?

```
# include <stdio.h>
# include <stdlib.h>

int main() {
    int a = 2;
    int b;
    int c = a + b;
    printf("valor de c: %i\n", c);
    return 0;
}
```

Estrutura condicional, leitura de string

/* Programa para avaliar o valor das cartas: Exemplo de else/else if, leitura de strings e conversão de variáveis */

```
# include <stdio.h>
```

```
# include <stdlib.h>
```

```
int main() {
```

```
    int card_num;
```

// número indica posição da carta

```
    puts("Digite o número da carta: ");
```

```
    scanf("%d", &card_num);
```

// Leitura

```
    int val = 0;
```

// variáveis podem ser declaradas em qualquer ponto do programa

```
    if (card_num == 0)
```

```
        val = 10;
```

```
    else if (card_num == 1)
```

// teste de uma nova condição

```
        val = 8;
```

```
    else if (card_num == 2)
```

// comparação é feita com ==

```
        val = 6;
```

```
    else if (card_num == 3)
```

```
        val = 12;
```

```
    else {
```

// caso contrário....

```
        val = - 1 ;
```

```
    }
```

```
    printf("O valor da carta %d é: %d\n", card_num, val);
```

// printf com mais de um argumento

```
    return 0;
```

```
}
```

Estrutura condicional, leitura de char

```
/* Programa para avaliar o valor das cartas: Exemplo de else/else if, leitura de strings e conversão de variáveis
*/

# include <stdio.h>
# include <stdlib.h>

int main() {
    char card_name[1];
    puts("Digite o nome da carta: ");
    scanf("%1s", card_name);
    int val = 0;
    if (card_name[0] == 'K')
        val = 10;
    else if (card_name[0] == 'Q')
        val = 8;
    else if (card_name[0] == 'J')
        val = 6;
    else if (card_name[0] == 'A')
        val = 12;
    else {
        val = atoi(card_name);
        val = val - 1 ;
    }
    printf("O valor da carta %s é: %i\n", card_name, val);
    return 0;
}
```

// string: array de caracteres, com 1 posição

// Leitura da string.

// variáveis podem ser declaradas em qualquer ponto do programa

// componentes do array são acessados por [posição]

// teste de uma nova condição

// caso contrário....

// conversão de string para inteiro

// printf com mais de um argumento

Estrutura condicional

/* Programa para avaliar o valor das cartas: Exemplo de else/else if, leitura de strings e conversão de variáveis */

```
# include <stdio.h>
```

```
# include <stdlib.h>
```

```
int main() {
```

```
    char card_name[2];
```

```
    puts("Digite o nome da carta: ");
```

```
    scanf("%10s", card_name);
```

```
    int val = 0;
```

```
    if (card_name[0] == 'K')
```

```
        val = 10;
```

```
    else if (card_name[0] == 'Q')
```

```
        val = 8;
```

```
    else if (card_name[0] == 'J')
```

```
        val = 6;
```

```
    else if (card_name[0] == 'A')
```

```
        val = 12;
```

```
    else {
```

```
        val = atoi(card_name);
```

```
        val = val - 1 ;
```

```
    }
```

```
    printf("O valor da carta %s é: %i\n", card_name, val);
```

```
    return 0;
```

```
}
```

O que aconteceria se:
char card_name[1] e
scanf("%2s", card_name)

// caso contrário....
// conversão de string para inteiro

// printf com mais de um argumento

Estrutura condicional, leitura de string

/* Programa para avaliar o valor das cartas: Exemplo de else/else if, leitura de strings e conversão de variáveis */

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main() {
```

```
    char card_name[2];
```

```
    puts("Digite o nome da carta: ");
```

```
    scanf("%1s", card_name);
```

```
    int val = 0;
```

```
    if (card_name[0] == 'K')
```

```
        val = 10;
```

```
    else if (card_name[0] == 'Q')
```

```
        val = 8;
```

```
    else if (card_name[0] == 'J')
```

```
        val = 6;
```

```
    else if (card_name[0] == 'A')
```

```
        val = 12;
```

```
    else {
```

```
        val = atoi(card_name);
```

```
        val = val - 1 ;
```

```
    }
```

```
    printf("O valor da carta %s é: %i\n", card_name, val);
```

```
    return 0;
```

```
}
```

Quando é necessário usar {}?

ponto do programa
por [posição]

// teste de uma nova condição


// caso contrário....

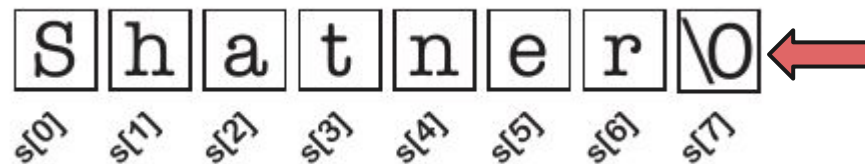
// conversão de string para inteiro

// printf com mais de um argumento

Array de caracteres

- A linguagem C trata strings como array de caracteres

`s = "Shatner"`  `s = {'S', 'h', 'a', 't', 'n', 'e', 'r'}`



sentinela: fim da string

- Existem bibliotecas que estendem a linguagem para fornecer funcionalidades de strings

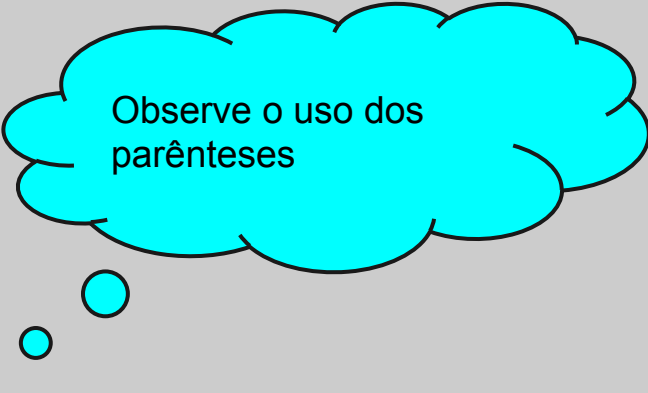
Operadores lógicos

and	&&	True se e somente se ambos os lados são True
and	&	idem acima, mas sempre avalia os dois lados
or		True se um dos lados é True
or		idem acima, mas sempre avalia os dois lados
not	!	Inverte o valor booleano

Operadores lógicos

```
# include <stdio.h>
# include <stdlib.h>

int main() {
    char card_name[3];
    puts("Enter the card_name: ");
    scanf("%2s", card_name);
    int val = 0;
    if (card_name[0] == 'K') {
        val = 1;
    } else if (card_name[0] == 'Q') {
        val = 8;
    } else if (card_name[0] == 'J') {
        val = 6;
    } else if (card_name[0] == 'A') {
        val = 12;
    } else {
        val = atoi(card_name);
    }
    if ((val > 2) && (val < 7))                // Verifica se valor está entre 3 e 6
        puts("Jack");
    else if ((val > 6) || ((val > 0) && (val < 2))) // caso contrário, verifica se carta é Q, A, ou K
        puts("Queen, King ou As");
    else
        puts("Carta Inválida");
    return 0;
}
```



Observe o uso dos parênteses

O que será escrito?

A

```
#include <stdio.h>

int main()
{
    int card = 1;
    if (card > 1)
        card = card - 1;
    if (card < 7)
        puts("Small card");
    else {
        puts("Ace!");
    }
    return 0;
}
```

B

```
#include <stdio.h>

int main()
{
    int card = 1;
    if (card > 1) {
        card = card - 1;
        if (card < 7)
            puts("Small card");
    }
    else
        puts("Ace!");
    return 0;
}
```

Verifique se cada código compila. Se sim, qual a saída?

C

```
#include <stdio.h>

int main()
{
    int card = 1;
    if (card > 1) {
        card = card - 1;
        if (card < 7)
            puts("Small card");
    } else
        puts("Ace!");

    return 0;
}
```

D

```
#include <stdio.h>

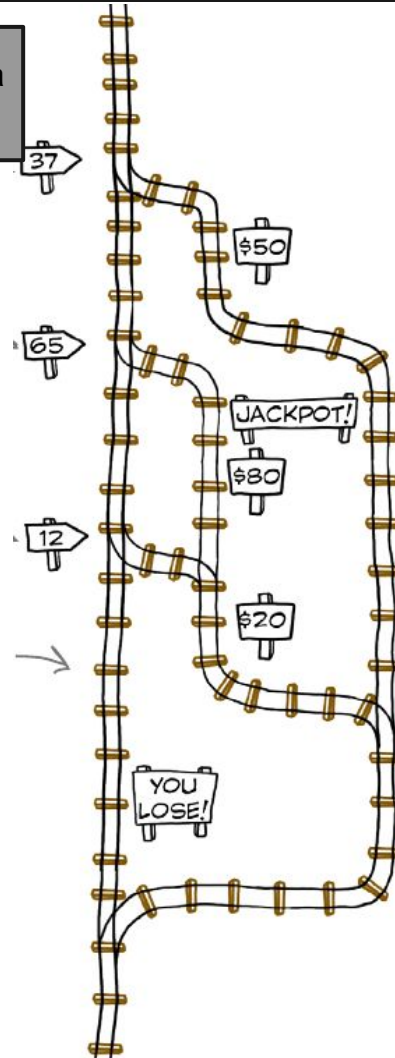
int main()
{
    int card = 1;
    if (card > 1) {
        card = card - 1;
        if (card < 7)
            puts("Small card");
    }
    else
        puts("Ace!");

    return 0;
}
```

Evitando uma enxurrada de ifs

```
switch (train) {  
  case 37: se entrar aqui, aumenta  
winnings de 50 e sai  
    winnings += 50;  
    break;  
  case 65:  
    puts("Jackpot!");  
    winnings += 80;  
  case 12:  
    winnings += 20;  
    break;  
  default:  
    winnings = 0;  
}
```

mas aqui,
ainda vai
entrar no
case 12 e
aumentar
winnings de
20



- testa uma única variável, procurando por casos que correspondam ao seu valor
- executa o código restante, até ver um break

Reescrava, usando switch

```
int val = 0;
if (card_name[0] == 'K') {
    val = 10;
} else if (card_name[0] == 'Q') {
    val = 10;
} else if (card_name[0] == 'J') {
    val = 10;
} else if (card_name[0] == 'A') {
    val = 11;
} else {
    val = atoi(card_name);
}
```


Loops

```
int counter = 1;
while (counter < 11) {
    printf("%i\n", counter);
    counter ++
}
```

- Testa a condição
- Se True, executa o corpo do while
- quando chega no fim do corpo, volta para a condição
- Lembre de inserir algum comando que torne a condição inválida em algum momento

```
int counter;
for (counter = 1; counter < 11; counter ++)  
    printf("%i\n", counter);
```

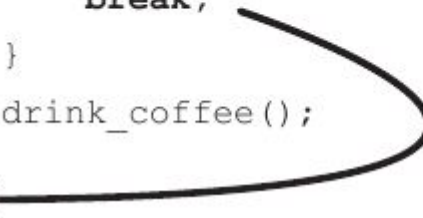
- inicializa a variável
- testa condição antes de entrar no corpo
- executa o corpo
- executa último comando na declaração do for, depois que sai do corpo
- variável de controle da iteração tem o valor da condição após o for

```
int counter = 1;
do {  
    printf("%i\n", counter);  
    counter ++  
} while (counter < 11);
```

- checa a condição do loop **depois** que executa comandos do corpo
- Sempre executa o corpo do loop ao menos uma vez


Alterando o fluxo de execução em um loop

```
while(feeling_hungry) {  
    eat_cake();  
    if (feeling_queasy) {  
        /* Break out of the while loop */  
        break;  
    }  
    drink_coffee();  
}
```



Sai imediatamente do loop mais interno, pulando o que estiver depois dele no corpo do loop

```
while(feeling_hungry) {  
    if (not_lunch_yet) {  
        /* Go back to the loop condition */  
        continue;  
    }  
    eat_cake();  
}
```



retorna ao início do loop, pulando comandos que estão depois dele

Constantes

- Para evitar usar números mágicos em um programa, sem dar uma pista do seu significado

```
/* Programa que imprime uma tabela de conversão Fahrenheit-Celsius */

#include <stdio.h>

#define LOWER 0      /* Limite inferior */
#define UPPER 300    /* Limite superior */
#define STEP 20      /* tamanho da passada */

/* Imprime Tabela Fahrenheit-Celsius */
int main() {
    int fahr;
    for (fahr = LOWER; fahr <= UPPER; fahr = fahr + STEP)
        printf("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
    return 0;
}
```

Exercício

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 0;
```

```
    int y = 0;
```

```
    while (x < 5) {
```



```
        printf("%i%i ", x, y);
```

```
        x = x + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Complete
com códigos
ao lado,
de acordo
com alguma
saída

```
y = x - y;
```

```
22 46
```

```
y = y + x;
```

```
11 34 59
```

```
y = y + 2;
```

```
if (y > 4)
```

```
    y = y - 1;
```

```
02 14 26 38
```

```
02 14 36 48
```

```
x = x + 1;
```

```
y = y + x;
```

```
00 11 21 32 42
```

```
if (y < 5) {
```

```
    x = x + 1;
```

```
    if (y < 3)
```

```
        x = x - 1;
```

```
}
```

```
y = y + 2;
```

```
11 21 32 42 53
```

```
00 11 23 36 410
```

```
02 14 25 36 47
```

Exercícios

1 - Escreva um programa em C que decide se um número é primo. Pergunte ao usuário qual número ele deseja verificar.

2 - Escreva um programa que pergunta ao usuário qual o valor do seu salário bruto e informa o valor a ser descontado de imposto de renda, de acordo com a tabela abaixo. A parcela a deduzir deve ser descontada da porcentagem de alíquota do IR para produzir o valor final a ser descontado.

Salário	Desconto	Parcela a deduzir
até 1499,14	isento	--
de 1499,15 a 2246,75	7,5 %	112,44
de 2246,76 a 2995,70	15 %	280,94
de 2995,71 a 3743,19	22,5 %	505,62
acima de 3743,19	27,5 %	692,78

Exercícios

3 - Escreva um programa que dados n números inteiros positivos, seja informada a soma dos números que são primos.

4 - Sabe-se que um número da forma n^3 é igual a soma de n ímpares consecutivos. Por exemplo: $1^3 = 1$, $2^3 = 3+5$, $3^3 = 7+9+11$, $4^3 = 13+15+17+19, \dots$ Escreva um programa que, dado um número m , sejam informados os ímpares consecutivos cuja soma é igual a n^3 para n assumindo valores de 1 a m . No exemplo acima, $m = 4$.

5 - Escreva um programa que lê números reais a , b e c , e calcula as raízes de uma equação do 2o grau da forma $ax^2 + bx + c = 0$. Caso a raiz seja complexa, informe a parte real e a parte imaginária.

Observação: Em C, para extrair raiz quadrada use a função `sqrt`, definida na biblioteca *math*.

Exercícios

6 - Escreva um programa em C que recebe como entrada um número inteiro e devolve o fatorial deste número

7 - Escreva um programa em C que caso um número informado seja primo, escreve os dois primos anteriores e os dois primos sucessores a ele

8 - Escreva um programa em C que imprima os N primeiros termos da série de Fibonacci, lembrando que esta série é definida da seguinte forma:

Termo 1 = 1

Termo 2 = 1

Termo $(x + 1) = \text{termo}(x) + \text{termo}(x - 1)$