Programação

Flavius Gorgônio
flavius@dct.ufrn.br

Agenda

- Estruturas de controle de repetição
- Estruturas com teste lógico
  - A estrutura while
  - A estrutura do...while
- Estruturas com contador
  - A estrutura for
- Outros mecanismos de controle de laços
  - Break e continue
  - Desvios incondicionais
- Estruturas de repetição aninhadas
- Exemplos de estruturas de repetição em programas
- Funções de randomização

Estruturas de repetição

Forma genérica da estrutura enquanto

Sintaxe genérica:

```
...
enquanto <condição> início
    <bloco_de_comandos>
fim
...
```

Semântica:

A condição (cor vermelha) é verificada e, no caso de ser verdadeira, o bloco de comandos subordinado à estrutura (cor azul) será executado

Ao final da execução do bloco, o controle do programa retorna ao teste e a condição é novamente verificada

Enquanto a condição for verdadeira, o bloco de comandos (chamado de laço) continuará a ser executado

Se a condição falhar, a execução continua após o término do laço

A estrutura while na linguagem C

```
...

while (<condição>) {

    instrução_1;

    instrução_2;

    ...

    instrução_n;

}

...
```

A instrução é denominada **`while()`**

O trecho de código entre chaves será repetido enquanto a condição descrita entre parênteses for diferente de zero

A condição de parada do laço deve ser sempre colocada entre parênteses e deve resultar em um valor inteiro

Operadores relacionais em C resultam em 0 (falso) ou 1 (verdadeiro)

As chaves são opcionais apenas quando uma única instrução estiver subordinada ao while, mas recomenda-se o seu uso em todos os casos

Exemplo de uso

```c
#include <stdio.h>

int main(void) {
  int i;
  i = 1;                  // inicialização
  while (i <= 10) {       // teste
    printf("Valor de i = %d\n",i);
    i = i + 1;            // incremento
  }
  printf("Fim do Programa!\n");
  return 0;
}
```

O laço while contém três partes:

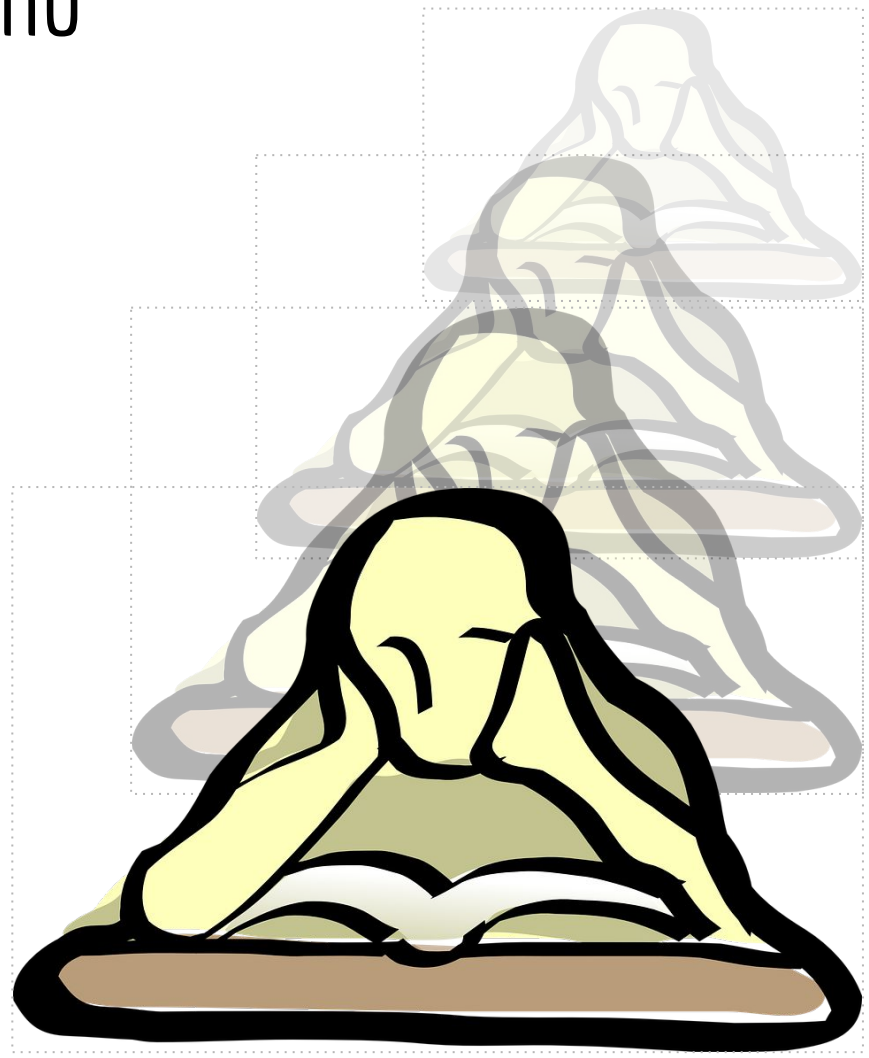
- Inicialização: onde a variável de controle do laço recebe um valor inicial
- Teste: onde é verificada a condição de parada do laço
- Incremento (ou decremento): localizado dentro do laço, altera o valor da variável de controle

https://repl.it/@flaviusgorgonio/exemploDeWhilec#main.c

Programa Média de Aluno

Escreva um programa na linguagem C que leia um conjunto de notas, calcule a média dessas notas e determine se o aluno foi APROVADO (média ≥ 7.0) ou REPROVADO (média < 7.0)

A quantidade de notas deve ser informada pelo usuário no início do programa



Programa Média de Aluno - Versão 1

```c
////////////////////////////////////////
// Programa Cálculo da Média de Aluno
// com Três Notas
// UFRN/DCT/DCT1106 - Programação
// by @flgorgonio - Mar/2019
////////////////////////////////////////
#include <stdio.h>
int main(void) {
 float n1, n2, n3, media;
 printf("Informe três notas:\n");
 printf("Nota 1: ");
 scanf("%f", &n1);
 printf("Nota 2: ");
 scanf("%f", &n2);
 printf("Nota 3: ");
 scanf("%f", &n3);
 media = (n1 + n2 + n3) / 3;
 printf("Sua média foi: %.1f\n", media);
 if (media >= 7.0) {
   printf("Parabéns, foi aprovado!\n");
 } else {
   printf("Pena, você foi reprovado!\n");
 }
 printf("Fim do Programa\n");
 return 0;
}

// Esta versão lê as três notas
// individualmente e calcula a média,
// mas se o número de notas aumentar
// será necessário incluir mais
// variáveis e funções de entrada e saída
```

https://repl.it/@flaviusgorgonio/mediaDoAlunoV1c#main.c

Programa Média de Aluno - Versão 2

```c
#include <stdio.h>
int main(void) {
 int i;
 float nota, somaNota, media;
 printf("Informe três notas:\n");
 somaNota = 0;
 i = 1;
 while (i <= 3) {
   printf("Informe a nota %d: ",i);
   scanf("%f", &nota);
   somaNota = somaNota + nota;
   i = i + 1;
 }
 media = somaNota / 3;
 printf("Sua média foi: %.1f\n", media);
 if (media >= 7.0) {
   printf("Parabéns, foi aprovado!\n");
 } else {
   printf("Pena, você foi reprovado!");
 }
 printf("Fim do Programa\n");
 return 0;
}

// Esta versão também lê três notas,
// mas utiliza um laço que pode ser
// redimensionado se o número de
// notas aumentar, não sendo mais
// necessário incluir outras variáveis
```

https://repl.it/@flaviusgorgonio/mediaDoAlunoV2c#main.c

Programa Média de Aluno - versão 3

```c
#include <stdio.h>
int main(void) {
 int i, n;
 float nota, somaNota, media;
 printf("Informe a quant. de notas:");
 scanf("%d", &n);
 somaNota = 0;
 i = 1;
 while (i <= n) {
   printf("Informe a nota %d: ",i);
   scanf("%f", &nota);
   somaNota = somaNota + nota;
   i = i + 1;
 }
 media = somaNota / n;
 printf("Sua média foi: %.1f\n", media);
 if (media >= 7.0) {
   printf("Parabéns, foi aprovado!\n");
 } else {
   printf("Pena, você foi reprovado!");
 }
 printf("Fim do Programa\n");
 return 0;
}

// Esta versão funciona com qualquer
// número de notas, bastando informar
// esse valor no início do programa
```

https://repl.it/@flaviusgorgonio/mediaDoAlunoV3c#main.c

Forma genérica da estrutura faça ... enquanto

Sintaxe genérica:

```
...
faça
    <bloco_de_comandos>
enquanto <condição>
...
```

Semântica:

O bloco de comandos subordinado à estrutura (cor azul) é executado uma primeira vez

Ao final do bloco, a condição (cor vermelha) é avaliada e, no caso de ser verdadeira, o bloco de comandos será executado novamente

Ao final de cada execução do bloco, a condição é novamente verificada. Enquanto permanecer verdadeira, o bloco de comandos continuará a ser executado

Se a condição falhar, a execução prossegue analisando as instruções posteriores ao laço

A estrutura do ... while na linguagem C

```
...

do {

    instrução_1;

    instrução_2;

    ...

    instrução_n;

} while (<condição>);

...
```

A instrução é denominada **do...while()**

O trecho de código entre chaves será executado pelo menos uma vez e será repetido enquanto a condição descrita entre parênteses for diferente de zero

A condição de parada do laço deve ser sempre colocada entre parênteses e deve resultar em um valor inteiro

Operadores relacionais em C resultam em 0 (falso) ou 1 (verdadeiro)

As chaves são opcionais apenas quando uma única instrução estiver subordinada ao do ... while, mas recomenda-se o seu uso em todos os casos

Exemplo de uso

```c
#include <stdio.h>
int menuPrincipal(void);
void moduloAluno(void);
void moduloProfessor(void);
void moduloDisciplina(void);
int main(void) {
  int op;
  do {
    op = menuPrincipal();
    switch (op) {
      case 1:   moduloAluno();
                break;
      case 2:   moduloProfessor();
                break;
      case 3:   moduloDisciplina();
                break;
    }
  } while (op != 0);
  printf("The End\n");
  return 0;
}
```

```c
int menuPrincipal(void) {
  int op;
  printf("\n1.Aluno\n2.Professor\n3.Disciplina\n");
  printf("Escolha: ");
  scanf("%d", &op);
  return op;
}

void moduloAluno(void) {
  printf("Este é o Módulo Aluno\n");
}

void moduloProfessor(void) {
  printf("Este é o Módulo Professor\n");
}

void moduloDisciplina(void) {
  printf("Este é o Módulo Disciplina\n");
}
```

https://repl.it/@flaviusgorgonio/menuComDoWhilec#main.c

Comparando estruturas: while versus do ... while

A estrutura while testa
ANTES de executar o bloco

A estrutura do ... while testa
APÓS executar o bloco

Forma genérica da estrutura para

Sintaxe genérica:

```
...
para <var> de <inicial> até <final> faça
     <bloco_de_comandos>
fim
...
```

Semântica:

O bloco de comandos subordinado à estrutura (cor azul) é repetido por um certo número de vezes, cujo valor é determinado pelos valores iniciais e finais do laço

A princípio, o valor <inicial> é atribuído à variável <var> e, caso esta não tenha atingido o valor <final>, o bloco de comando será executado

A cada iteração do laço, a variável <var> tem seu valor incrementado (ou decrementado). Enquanto a variável <var> não atingir o valor <final>, o bloco de comandos continuará a ser executado

Ao final, a execução prossegue analisando as instruções posteriores ao laço

A estrutura for na linguagem C

```
...

for (<inic>;<teste>;<incr/decr>) {

    instrução_1;

    instrução_2;

    ...

    instrução_n;

}

...
```

Em C, a instrução é denominada **`for()`**

Antes da primeira iteração, a <inicialização> é executada. O <teste> é então realizado e o trecho de código entre chaves será executado sempre que a condição de <teste> for diferente de zero.

Esse teste deve resultar em um valor inteiro. Operadores relacionais em C resultam em 0 (falso) ou 1 (verdadeiro)

Ao final de cada iteração, o <incremento> ou <decremento> é executado

As chaves são opcionais apenas quando uma única instrução estiver subordinada ao for, mas recomenda-se o seu uso em todos os casos

Exemplos de uso

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

int main(void) {
  int k;
  srand(time(NULL));
  for (int i = 0; i < 10; i++) {
    k = rand() % 10;
    printf("valor = %d\n", k);
  }
  return 0;
}
```

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

void delay(int segundos) {
  int milissegundos = 1000 * segundos;
  clock_t inicio = clock();
  while (clock() < inicio + milissegundos)
    ;
}

int main() {
  int i;
  for (i = 10; i > 0; i--) {
    printf("%d...\n", i);
    delay(1);
  }
  printf("Fogo!\n");
  return 0;
}
```

Escopo de variáveis em laços

Padrão C ANSI:

```c
int main(void) {
  int i;
  for (i = 0; i < 10; i++) {
    printf("valor = %d\n", i);
  }
  return 0;
}
```

O escopo da variável **`i`** estende-se desde a sua declaração até o final da função onde é declarada

Padrão C99:

```c
int main(void) {
  for (int i = 0; i < 10; i++) {
    printf("valor = %d\n", i);
  }
  return 0;
}
```

Nesse caso, o escopo da variável **`i`** limita-se ao laço onde é declarada e o espaço de memória ocupado pela mesma é devolvido ao sistema operacional ao final do laço

Mecanismos de controle de laços: break

```c
#include <stdio.h>

int primo(int);

int main(void) {
 int i, k, inf, sup, vet[10];
 k = 0;
 printf("Informe os limites do intervalo\n");
 scanf("%d %d", &inf, &sup);
 for (i = inf; i < sup; i++) {
   if (primo(i))
     vet[k++] = i;
   if (k == 10)
     break;
 }
 printf("Os 10 primeiros primos do
intervalo:\n");
 for (i = 0; i < k; i++)
   printf("%d\n", vet[i]);
 return 0;
}
```

```c
int primo(int n) {
  int i;
  if (n == 1)
    return 0;
  for (i = 2; i < n; i++) {
    if (n % i == 0)
      return 0;
  }
  return 1;
}
```

https://repl.it/@flaviusgorgonio/lacoComBreakc#main.c

Mecanismos de controle de laços: continue

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

int jaExiste(int, int[], int);

int main(void) {
 int i, k, v, existe, vet[10];
 srand(time(NULL));
 printf("Quantos números deseja no cartão? ");
 scanf("%d", &k);
 i = 0;
 while (i < k) {
   v = 1 + rand() % 60;
   printf("sorteio = %d\n", v);
   existe = jaExiste(v, vet, k);
   if (existe)
     continue;
   vet[i] = v;
   i++;
 }
 printf("Os números do seu cartão são:\n");
 for (i = 0; i < k; i++)
   printf("%d\t", vet[i]);
 return 0;
}

int jaExiste(int v, int vet[], int k) {
 int i;
 for (i = 0; i < k; i++) {
   if (v == vet[i])
     return 1;
 }
 return 0;
}
```

https://repl.it/@flaviusgorgonio/lacoComContinuec#main.c

Desvio incondicional

Uma sentença de desvio incondicional transfere o controle da execução para uma posição especificada no programa

Muitas linguagens de programação incluem esse recurso, apesar da maioria dos cientistas e projetistas de linguagens considerá-lo perigoso

Apesar de bastante flexível, tal recurso pode tornar os programas difíceis de serem lidos e mantidos, devendo portanto ser evitado

Por isso, é sempre preferível usar as estruturas de controle tradicionais

Java, Ruby e Python, por questões de segurança, não possuem esse recurso

```c
#include <stdio.h>

int main(void) {
  int i;
  printf("Laço com goto\n");
  i = 1;
  loop:
    if (i > 10)
      goto fora;
    printf("%d\n", i);
    i += 1;
    goto loop;
  fora:
    printf("Fim do Programa!\n");
  return 0;
}
```

https://repl.it/@flaviusgorgonio/lacoComGotoc#main.c

Laços aninhados com for

```c
#include <stdio.h>

int main(void) {
 printf("i\tj\n");
 printf("=\t=\n");
 for (int i = 1; i <= 3; i++) {
   for (int j = 1; j <= 4; j++) {
     printf("%d\t%d\n", i, j);
   }
   printf("\n");
 }
 printf("Fim\n");
 return 0;
}
```

Resultado obtido:

```
i   j
=   =
1   1
1   2
1   3
1   4

2   1
2   2
2   3
2   4

3   1
3   2
3   3
3   4

Fim
```

https://repl.it/@flaviusgorgonio/lacosForAninhadosc#main.c

Laços aninhados com while

```c
#include <stdio.h>
int main(void) {
  char resp1, resp2;
  printf("Neste momento, você está antes do primeiro laço\n");
  printf("Você quer entrar no primeiro laço? ");
  scanf("%c", &resp1);
  getchar();
  while (resp1 == 's') {
    printf("\tVocê está dentro do primeiro laço\n");
    printf("\tVocê quer entrar no segundo laço? ");
    scanf("%c", &resp2);
    getchar();
    while (resp2 == 's') {
      printf("\t\tVocê está dentro do segundo laço\n");
      printf("\t\tVocê quer continuar no segundo laço? ");
      scanf("%c", &resp2);
      getchar();
    }
    printf("\tVocê está dentro do primeiro laço\n");
    printf("\tVocê quer continuar no primeiro laço? ");
    scanf("%c", &resp1);
    getchar();
  }
  // continuação
  printf("Você saiu do programa\n");
  return 0;
}
```

https://repl.it/@flaviusgorgonio/lacosWhileAninhadosc#main.c

Busca por divisores de um número

```c
#include <stdio.h>
int main(void) {
  int num;
  printf("Divisores de um número\n");
  printf("Informe um número inteiro: ");
  scanf("%d", &num);
  printf("Os divisores de %d são:\n", num);
  for (int i = 1; i <= num; i++) {
    if (num % i == 0) {
      printf("%d\t", i);
    }
  }
  printf("\n");
  return 0;
}
```

https://repl.it/@flaviusgorgonio/DivisoresDeUmNumeroc-1#main.c

```c
#include <stdio.h>
int main(void) {
  int num;
  int primeiro, ultimo;
  printf("Encontra divisores de vários números\n");
  printf("Informe o valor inicial: ");
  scanf("%d", &primeiro);
  printf("Informe o valor final: ");
  scanf("%d", &ultimo);
  for (int num = primeiro; num <= ultimo; num++) {
    printf("Os divisores de %d são:\n", num);
    for (int i = 1; i <= num; i++) {
      if (num % i == 0) {
        printf("%d\t", i);
      }
    }
    printf("\n");
  }
  return 0;
}
```

https://repl.it/@flaviusgorgonio/DivisoresDeUmNumeroc-3#main.c

Busca por números primos

```c
#include <stdio.h>
int main(void) {
  int num, cont;
  cont = 0;
  printf("Divisores de um número\n");
  printf("Informe um número inteiro: ");
  scanf("%d", &num);
  printf("Os divisores de %d são:\n", num);
  for (int i = 1; i <= num; i++) {
    if (num % i == 0) {
      cont++;
      printf("%d\t", i);
    }
  }
  printf("\n");
  if (cont == 2) {
    printf("O número %d é primo.\n", num);
  } else {
    printf("O número %d não é primo.\n", num);
  }
  return 0;
}
```

https://repl.it/@flaviusgorgonio/DivisoresDeUmNumeroc-2#main.c

```c
#include <stdio.h>
int main(void) {
  int num, cont;
  int primeiro, ultimo;
  printf("Encontra primos em um intervalo\n");
  printf("Informe o valor inicial: ");
  scanf("%d", &primeiro);
  printf("Informe o valor final: ");
  scanf("%d", &ultimo);
  for (int num = primeiro; num <= ultimo; num++) {
    cont = 0;
    printf("Os divisores de %d são:\n", num);
    for (int i = 1; i <= num; i++) {
      if (num % i == 0) {
        cont++;
        printf("%d\t", i);
      }
    }
    printf("\n");
    // completar como no exemplo anterior
  }
  return 0;
}
```

https://repl.it/@flaviusgorgonio/DivisoresDeUmNumeroc-4#main.c

Busca por números primos

```c
#include <stdio.h>

int main(void) {
  int num, cont;
  int primeiro, ultimo;
  printf("Encontra primos em um intervalo\n");
  printf("Informe o valor inicial: ");
  scanf("%d", &primeiro);
  printf("Informe o valor final: ");
  scanf("%d", &ultimo);
  printf("Os primos entre %d e %d são:\n",
         primeiro, ultimo);
  for (int num = primeiro; num <= ultimo; num++) {
    cont = 0;
    for (int i = 1; i <= num; i++) {
      if (num % i == 0) {
        cont++;
      }
    }
    if (cont == 2) {
      printf("%d\t", num);
    }
  }
  return 0;
}

// modifique esse programa incluindo uma função
// que verifica se um número é ou não primo,
// retornando 1 ou 0
```

https://repl.it/@flaviusgorgonio/DivisoresDeUmNumeroc-5#main.c

Números primos em um intervalo

```c
#include <stdio.h>
#include <math.h>
int primo1(int);
int primo2(int);
int primo3(int);
int main(void) {
  int primeiro, ultimo;
  printf("Encontra números primos em um intervalo\n");
  printf("Informe o valor inicial: ");
  scanf("%d", &primeiro);
  printf("Informe o valor final: ");
  scanf("%d", &ultimo);
  printf("Os primos entre %d e %d são:\n", primeiro, ultimo);
  for (int num = primeiro; num <= ultimo; num++) {
    if (primo1(num)) {        // testar com as três funções
      printf("%d\t", num);
    }
  }
  return 0;
}
```

```c
int primo1(int n) {
  int cont = 0;
  for (int i = 1; i <= n; i++) {
    if (n % i == 0) {
      cont++;
    }
  }
  if (cont == 2) {
    return 1;
  } else {
    return 0;
  }
}
```

https://repl.it/@flaviusgorgonio/buscaPrimosc#main.c

Função que verifica primos

```c
int primo2(int n) {
  int cont = 0;
  if (n == 1) {
    return 0;
  }
  for (int i = 2; i < n; i++) {
    if (n % i == 0) {
      cont++;
    }
  }
  if (cont == 0) {
    return 1;
  } else {
    return 0;
  }
}
```

```c
int primo3(int n) {
  int i = 2;
  int raiz = sqrt(n);
  if (n == 1) {
    return 0;
  }
  while (i <= raiz) {
    if (n % i == 0) {
      return 0;
    }
    i++;
  }
  return 1;
}
```

https://repl.it/@flaviusgorgonio/buscaPrimosc#main.c

Programa Cara ou Coroa

Escreva um programa em C que simule uma partida do jogo "Cara ou Coroa" entre um jogador humano e um computador.

No início da partida, o jogador deve optar por "Cara" ou "Coroa", sendo que o computador fica com a opção remanescente.

Ao final da partida, o programa deve informar ao usuário quem foi o vencedor.

Programa Cara ou Coroa

```
//////////////////////////////////////
// Use o trecho de código abaixo como
// exemplo para a sua implementação
//////////////////////////////////////
#include <stdlib.h>
...
moeda = rand() % 2;
if ((jog == 0) && (moeda == 0))
  printf("Deu Cara, você ganhou!\n");
else if ((jog == 0) && (moeda == 1))
  printf("Deu Coroa, você perdeu!\n");
else if ((jog == 1) && (moeda == 1))
  printf("Deu Coroa, você ganhou!\n");
else
  printf("Deu Cara, você perdeu!\n");
...
```

Algumas dicas:

A face da moeda (cara ou coroa) deve ser definida a partir do gerador de números pseudo-aleatórios da linguagem

A função **`rand()`** está definida na biblioteca <stdlib.h>

A função **`rand()`** gera um valor (pseudo) aleatório entre 0 e RAND_MAX (constante definida na biblioteca <stdlib.h>)

https://repl.it/@flaviusgorgonio/caraOuCoroac#main.c

Compreendendo melhor a função rand()

Algumas considerações:

Os números gerados estão no intervalo entre 0 e a constante **RAND_MAX**, cujo valor é definido no arquivo **<stdlib.h>**

A função retorna um valor do tipo float

Usando o operador módulo (%) é possível convertê-lo em um int, dentro do intervalo desejado

Execute o programa ao lado e veja o gerador de números aleatórios em ação. repita o processo e veja que os números seguem a mesma sequência

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

int main(void) {
  int k;
  for (int i = 0; i < 10; i++) {
    k = rand() % 10;
    printf("valor = %d\n", k);
  }
  return 0;
}
```

```c
// Gera números inteiro dentro do
// intervalo [0,9]
```

Compreendendo melhor a função rand()

A função **`rand()`** não gera números realmente aleatórios, mas pseudo aleatórios, que seguem um conjunto de valores previamente definido

Os números gerados seguem sempre a mesma sequência, pois são obtidos a partir de uma tabela

É possível usar outra função, **`srand()`**, para inicializar o gerador com uma "semente", que define de que ponto da tabela parte a sequência

Costuma-se iniciar a semente com o valor da função **`time(NULL)`**, que retorna o total de segundos passados desde 1 de janeiro de 1970 e a data atual, uma semente diferente a cada execução

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

int main(void) {
  int k;
  srand(time(NULL));
  for (int i = 0; i < 10; i++) {
    k = rand() % 10;
    printf("valor = %d\n", k);
  }
  return 0;
}
```

Eficácia do gerador de números aleatórios

Será que o gerador de números aleatórios usado na função rand() é realmente eficiente?

Modifique o programa Cara ou Coroa de forma que ele permita disputar 100 partidas seguidas. Compute o número de vitórias e derrotas do jogador humano. O ideal é esse valor fique próximo a 50%

Modifique-o novamente de forma que ele permita disputar n partidas, onde o valor de n é informado pelo usuário no início do jogo. Como o gerador de números aleatório se comporta com grandes valores de n?

Gere um dado aleatório e compute quantas vezes (em valores percentuais) cada face do dado é sorteada se o programa for executado 10, 100, 1.000, 10.000 e 100.000 vezes

https://repl.it/@flaviusgorgonio/moedaViciadac#main.c

Um dado digital é realmente honesto?

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

int main(void) {
  int i, qde, dado;
  int   um = 0, dois = 0;
  int tres = 0, quat = 0;
  int cinc = 0, seis = 0;
  srand(time(NULL));
  i = 0;
  printf("Quantos sorteios? ");
  scanf("%d", &qde);
  while (i < qde) {
    dado = 1 + rand() % 6;
    i++;
    switch (dado) {
      case 1: um++;
          break;
      case 2: dois++;
          break;
      case 3: tres++;
          break;
      case 4: quat++;
          break;
      case 5: cinc++;
          break;
      case 6: seis++;
          break;
     }
  }
  printf("\nFace \t Vezes \t    %%\n");
  printf(" 1\t  %d \t %7.2f %%\n", um,
(um*100.0/i));
  printf(" 2\t  %d \t %7.2f %%\n", dois,
(dois*100.0/i));
  printf(" 3\t  %d \t %7.2f %%\n", tres,
(tres*100.0/i));
  printf(" 4\t  %d \t %7.2f %%\n", quat,
(quat*100.0/i));
  printf(" 5\t  %d \t %7.2f %%\n", cinc,
(cinc*100.0/i));
  printf(" 6\t  %d \t %7.2f %%\n", seis,
(seis*100.0/i));
  return 0;
}
```

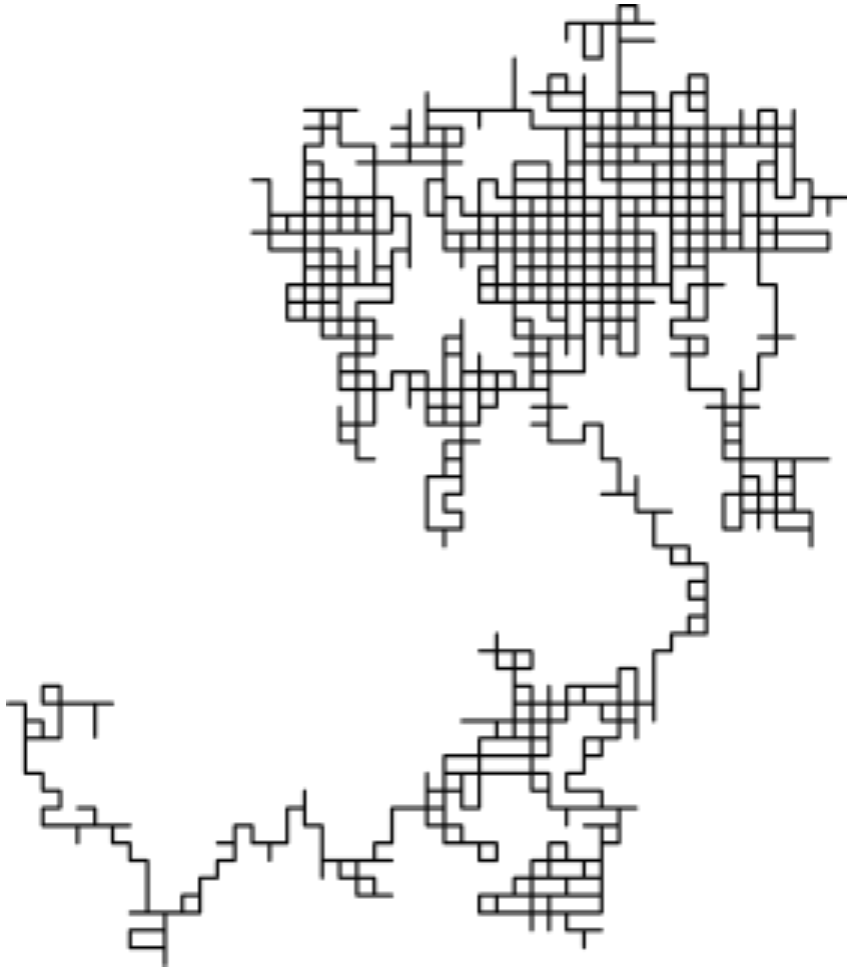
https://repl.it/@flaviusgorgonio/dadoViciadoc#main.c

Passeio aleatório (*random walk*)

Um passeio aleatório (*random walk*) é um modelo matemático que descreve um caminho composto por uma sucessão de passos aleatórios

Diversos fenômenos naturais podem ser descritos, representados, aproximados ou simulados por um passeio aleatório

Alguns exemplos incluem o caminho traçado por uma molécula em um líquido ou um gás, o trajeto de um animal buscando alimento, o preço flutuante de ações ou a situação financeira de um jogador



Simulando um passeio aleatório

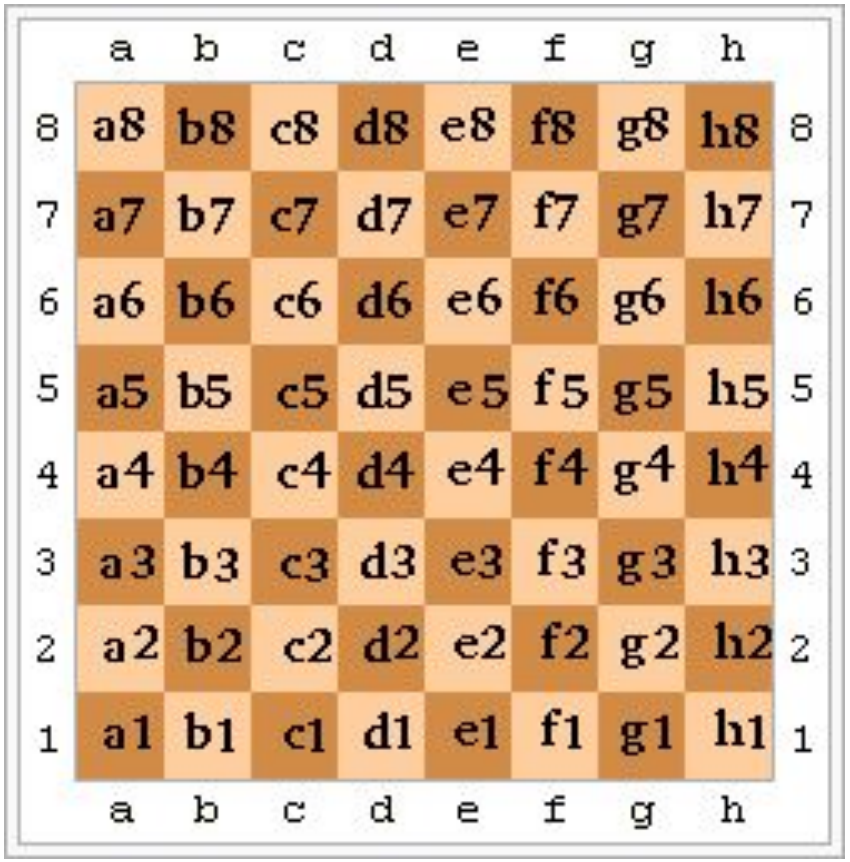
Suponha que uma peça sai da posição **a1** e deseja atingir a posição **h8** em um tabuleiro de xadrez conforme representado ao lado

Em cada movimento, a peça pode mover-se para a esquerda, para a direita, para cima ou para baixo, mas apenas uma casa de cada vez. Movimentos além da borda do tabuleiro não são permitidos

Escreva um programa em C que simule um passeio aleatório dessa natureza, descrevendo o caminho da peça do ponto de partida até o ponto final desejado

Verifique quantos movimentos foram necessários e compare seu resultado com os obtidos por seus colegas

Notação algébrica para tabuleiro de xadrez



Jogo de dados: 7 ou 11

Escreva um programa em linguagem C que simule um jogo de dados disputado entre um jogador humano e um computador, onde dois dados eletrônicos (simulados por software, através de valores aleatórios) devem ser lançados simultaneamente

O jogador vence se a soma dos pontos dos dois dados for 7 ou 11, caso contrário vence o computador

Ao final da partida, o programa deverá perguntar ao usuário se o mesmo deseja jogar novamente. O programa deverá permitir uma nova partida, caso a resposta seja afirmativa ou encerrar em caso negativo

Ao final do jogo, o programa deve apresentar o resultado final

Jogo do Zerinho ou Um

Escreva um programa em linguagem C que simule o tradicional jogo do ZERINHO ou UM

Nesse jogo, deve haver pelo menos três jogadores, onde o primeiro é um jogador humano e os demais são simulados pelo computador

Para jogar, cada participante humano (pode haver mais de um) deve escolher um dos valores: 0 (ZERO) ou 1 (UM). Vence aquele que apresentar um valor distinto de todos os outros

Se todos escolherem números iguais, a partida está empatada. O programa deve indicar se houve um vencedor ou se houve empate

Permita ao jogador repetir o jogo, caso deseje, ou continuar jogando

Jogo do pedra, papel e tesoura

Escreva um programa em linguagem C que simule o jogo PEDRA, PAPEL e TESOURA, a ser disputado entre um jogador humano e o computador

O jogador humano deverá escolher entre uma das três opções e a escolha do computador deverá ser feita de forma aleatória

O programa deverá realizar o julgamento e definir quem venceu o jogo, lembrando que PEDRA vence TESOURA, TESOURA vence PAPEL e PAPEL vence PEDRA. Considere a possibilidade de haver empate

O jogador deverá poder jogar novamente, caso deseje, ou continuar a disputar outras partidas. Ao final, o programa deve apresentar uma tabela com resultados

Utilize uma representação gráfica, com caracteres, para as figuras do jogo

Adivinhe o número

Escreva um programa na linguagem C que simule um jogo de adivinhação, onde o computador sorteará um valor entre 1 e 100 e o jogador terá, no máximo, dez chances para acertar o número

Caso o usuário acerte, o programa deverá parabenizá-lo, em seguida, encerrar o programa, mostrando quantos palpites foram necessários para ele acertar

Se errar, o programa deverá fornecer uma dica, dizendo “DIGITE UM NÚMERO MENOR” ou “DIGITE UM NÚMERO MAIOR”, de acordo com o valor fornecido

Se o usuário exceder as dez tentativas, o programa deverá encerrar a partida e informar o motivo da sua desclassificação

Se o usuário informar um palpite fora da faixa permitida (considerando os valores que forem sendo informados), o programa também deverá desclassificá-lo

Por exemplo:

Informe um número: 50
Digite um número MENOR
Informe um número: 25
Digite um número MAIOR
Informe um número: 20
Valor fora da faixa, você foi DESCLASSIFICADO

Qual o seu time do coração?

Escreva um programa em linguagem C que permita entrevistar um grupo de pessoas sobre seu time de futebol preferido

Cada indivíduo entrevistado deverá escolher entre os 20 times que disputam a Série A do Campeonato Brasileiro

Não há um número predefinido de pessoas a serem entrevistadas e o programa deve ser fácil de usar o bastante para que os próprios usuários respondam à entrevista

Entretanto, o programa deverá conter mecanismos que evitem (ou tentem evitar) possíveis fraudes

Por exemplo, evite que a mesma pessoa tente votar várias vezes seguidas, ou interrompa a execução do programa, ou digite informações que travem o programa, etc.

Ao inicializar o programa, o operador irá definir por quanto tempo (em minutos) o programa deverá permanecer coletando informações

Ao final do tempo determinado pelo operador, o programa deverá encerrar a execução das entrevistas e apresentar o resultado final da votação

Bibliografia recomendada

CELES FILHO, W. *et al*. Introdução a estruturas de dados. Rio de Janeiro: Elsevier, 2004.

KERNIGHAN, B. W.; PIKE, R. The practice of programming. Boston: Addison-Wesley, c1999.

VAREJÃO, F. M. Linguagens de programação: conceitos e técnicas. Rio de Janeiro: Elsevier, 2004.