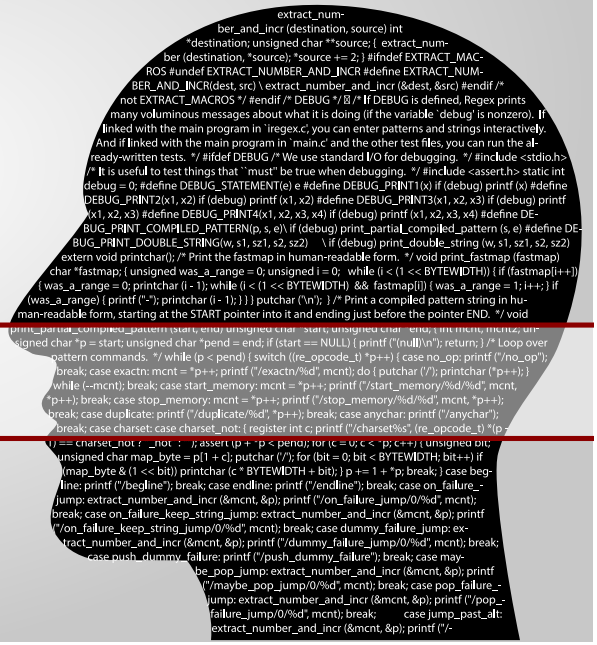
```
4   #include "utilities.h"
5
6   Game game;
7
8   int main(int argc, char **argv) {
9
10   game = Game();
11
12   game.initialize();
13  }
```



3. If-else-switch

Os comandos de decisão são muito úteis na programação de computadores. Neste momento, veremos a utilização de operadores relacionais e lógicos. Na linguagem C, existem dois tipos de operadores de decisão os blocos if-else e switch. O if-else permite avaliar se uma determinada expressão lógica (booleana) é verdadeira, então um bloco de código é executado. Opcionalmente, pode-se colocar um bloco de código para executar para o caso da expresão anterior for falsa.

3.1 Operadores lógicos e relacionais

Neste instante, é interessante que tenhamos em mente a respeito das operações da lógica booleana. Nesta lógica, temos dois valores possíveis (verdadeiro ou falso). Para o computador, em especial a linguagem C, tomaremos os seguintes conceitos:

- Valor **falso (F)** é **igual** a 0 (zero);
- Valor **verdadeiro (V)** é qualquer valor **diferente** de 0 (zero)

Diante do exposto, podemos, sem perda de generalização, assumir os valores de 1 para verdadeiro e 0 para falso, afinal estes valores atendem às definições acima.

Na lógica booleana, os operadores básicos são **NOT**, **AND** e **OR**. Suponhamos que existam duas expressões booleanas A e B. Neste caso, o valor resultante lógico dos operadores básicos está descrito na tabela 3.1.

Tabela 3.1: Tabelas-verdade para os operadores lógicos básicos.

A	B	A AND B	A OR B	NOT(A)
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

Tabela 3.3: Operadores da lógica booleana em linguagem C.

Operador	Explicação
&&	Operação de AND lógico só é verdadeiro quando todas as equações envolvidas forem verdadeiras
||	Operador lógico OR, só é falso quando todas as equações envolvidas forem falsas
!	Operador lógico NOT. Ele alterna o valor do operando. Caso o operando seja 1, ele muda para 0, e vice e versa.

Adaptando os operadores da tabela 3.3 na tabela 3.1, geramos a tabela 3.5.

Tabela 3.5: Tabelas-verdade para os operadores lógicos básicos.

A	B	A && B	A || B	!A
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

Com as operações definidas nas tabelas 3.3 e 3.7, facilmente se consegue representar as situações esperadas para controle de decisão em certos passos da computação, através de expressões booleanas.

Tabela 3.7: Operadores relacionais.

Operador	Explicação
>	Devolve o valor lógico correspondente à pergunta: é maior que?
>=	Devolve o valor lógico correspondente à pergunta: é maior ou igual que?
<	Devolve o valor lógico correspondente à pergunta: é menor que?
<=	Devolve o valor lógico correspondente à pergunta: é menor ou igual que?
==	Devolve o valor lógico correspondente à pergunta: é igual a?
!=	Devolve o valor lógico correspondente à pergunta: é diferente?

Lista 3.1 - Exercícios de fixação - operadores lógicos

Exercício 3.1 Determine o resultado lógico das expressões mencionadas, assinalando se são verdadeiras ou falsas. Considere para as repostas, os seguintes valores: X=1, A=3, B=5, C=8, D=7.

a) !(X > 3)
□ Verdadeira ou □ Falsa

b) (X < 1) && !(C > 5)
□ Verdadeira ou □ Falsa

c) !(D < 0) && (C > 5)
□ Verdadeira ou □ Falsa

d) !(X > 3) || (C<7)
□ Verdadeira ou □ Falsa

e) (A > B) || (C > B)
□ Verdadeira ou □ Falsa

f) (X>=2)
□ Verdadeira ou □ Falsa

g) (X < 1) && (B >= D)
□ Verdadeira ou □ Falsa

h) (D < 0) || (C > 5)
□ Verdadeira ou □ Falsa
i) !(D > 3) || !(B < 7)
□ Verdadeira ou □ Falsa
j) (A > B) || !(C > B)
□ Verdadeira ou □ Falsa

Exercício 3.2 Considerando as variáveis declaradas na tabela abaixo e mais a variável booleana TESTE, com valor 0, avalie as expressões a seguir, para cada uma das três combinações de valores apresentadas:

Situação	A	B	NOME	PROFISSAO
I	3	4	'M'	'a'
II	5	8	'P'	'm'
III	2.5	3	'A'	'p'

a) (A + 1 >= B) || (NOME != 'A')
b) ((A + 1 >= B) && (PROFISSAO == 'm'))
c) (NOME != 'A') || (PROFISSAO == 'm') && (A + 1 >= B)
d) (!TESTE && (A + 1 >= B) || !(PROFISSAO == 'm'))
e) !(A + 1 >= B) && TESTE)

Preencha as lacunas com **verdadeiro** ou **falso**, dependendo do resultado das expressões:

Situação	a)	b)	c)	d)	e)
I					
II					
III					

3.2 Blocos de controle de decisão If-else

A construção de um trecho com if-else é simples, conforme mostrado a seguir.

Listing 3.1: Estrutura if-else

```
1 if(<expressao>)
2   Bloco_if
3 else//Opcional
4   Bloco_else;
```

Na listagem 3.1, o identificador `Bloco_if` só será executado caso a <expressao> retornar um valor verdadeiro. Cada if, dá o direito da um bloco else, para caso o valor da expressão booleana no momento da avaliação seja falso. Repare que desta forma, cada vez que o programa passar por este trecho, ou `Bloco_if` será executado ou `Bloco_else` será executado, mas **nunca ambos**.

Tanto o bloco de códigos do if quanto o bloco de códigos do else podem ter uma ou mais instruções. No caso de haver apenas uma instrução, *não há necessidade* do uso de chaves. Sobretudo, tendo mais de uma instrução, o bloco de códigos

deve estar dentro de abre e fecha chaves. Uma dica para quem está começando é: sempre use chaves!

Cuidado

Observe que ao final do if e do else não há ponto e vírgula. Caso você coloque um ponto e vírgula, este será entendido como a única instrução a ser executada caso o valor da expressão booleana seja verdadeiro.

Imagine que você está implementando o software para uma máquina de vendas de tickets. A máquina só pode aceitar notas de 20, 50 e 100 reais. Para isso, podemos montar uma expressão booleana que, dado a nota de entrada, verifica se é um destes casos. Desta forma, usando-se o if, uma expressão boleana possível para representar é a que se apresenta no Exemplo 3.1.

■ **Examplo 3.1**

```
1  #include<stdio.h>
2  int main(){
3    int nota;
4    puts("Entre com a nota: (100, 50 ou 20): ");
5    scanf("%d", &nota);
6    if(nota == 100 || nota == 50 || nota == 20){
7      //procede com o restante do algoritmo
8      puts("Muito bem. Você inseriu uma das notas suportadas");
9    }else{
10     puts("Nota não suportada por esta máquina.");
11   }
12   return 0;
13 }
```

■

Em alguns casos, se deseja fechar o programa quando uma das notas não é a aceitável. Neste caso, o que ocorreria se colocássemos a seguinte verificação:

```
1  #include<stdio.h>
2  int main(){
3    int nota;
4    puts("Entre com a nota: (100, 50 ou 20): ");
5    scanf("%d", &nota);
6    if(!(nota == 100 || nota == 50 || nota == 20)){
7      printf("Nota não suportada por esta máquina.");
8      return 0;
9    }
10   //procede com o restante do algoritmo
11   puts("Muito bem. Você inseriu uma das notas suportadas");
12   return 0;
13 }
```

Qualquer nota correta que o usuário colocar será avaliado como falso no if. E portanto, ele simplesmente pula para a instrução puts. Caso a nota não seja a que

se espera (100, 50 ou 20), o programa emite a mensagem e encerra (return 0 na função main faz o programa encerrar).

3.2.1 If-else aninhados

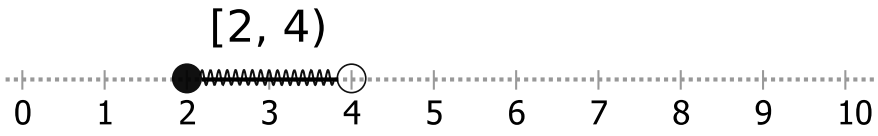


Figura 3.1: Exemplo de representação visual do invervalo $[2, 4)$. Perceba que o intervalo é fechado em 2, ou seja, o número 2 pertence ao intervalo, enquanto que é aberto em 4, ou seja, o número 4 não pertence ao intervalo.

Agora suponha que você tenha um problema de verificar se uma determinada nota está ou não em um intervalo. Considere o problema 3.2.

■ **Examplo 3.2** Faça um programa que receba três notas de um aluno, calcule e mostre a média aritmética e a mensagem constante na tabela a seguir. Aos alunos que ficaram para exame, calcule e mostre a nota que deverão tirar para serem aprovados, considerando que a média exigida é 5,0.

Média	Mensagem
$[0, 4)$	Reprovado
$[4, 7)$	Exame Final
$[7, 10]$	Aprovado

Representando os intervalos de forma visual, temos uma distribuição da forma da Figura 3.2.

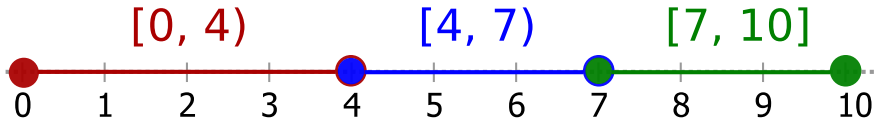


Figura 3.2: Representação dos intervalos do Problema 3.2.

Observando os intervalos apresentados na Figura 3.2, nos seguintes Listings:

Listing 3.2: Listagem com o if da mensagem Reprovado.

```
1 if(media>=0 && media <4){
2   puts("Reprovado");
3 }
```

Listing 3.3: Listagem com o if da mensagem Final.

```
1 if(media>=4 && media <7){
2   puts("Final");
3 }
```

Listing 3.4: Listagem com o if da mensagem Aprovado.

```
1 if(media>=7 && media <=10){
2   puts("Aprovado");
3 }
```

Ou seja, se colocarmos em um programa só, os três ifs, em um programa que toma a média (ou calcula como é o caso do problema), emitirá as mensagens corretamente. Sobretudo, quando se trata de intervalos, quando um determinado número não estiver em um intervalo, logo ele estará no complemento deste intervalo. Isto é retratado no else. Permita-me, então, utilizarmos a informação do else. Vamos assumir que a média está no intervalo $[0, 10]$. Neste caso, analise o seguinte código na Listagem 3.5:

Listing 3.5: Listagem com código alternativo.

```
1  #include<stdio.h>
2  int main(){
3    float media;
4    puts("Entre com a média:");
5    scanf("%f", &media);
6    if(media<4){
7    //Estamos reprovados
8      puts("Reprovado");
9    }else{ //se o código entrar aqui, significa que media >=4
10     if(media <7){
11       puts("Final");
12     }else{
13       puts("Aprovado");
14     }
15   }
16   return 0;
17 }
```

Repare que como se percebe na Listagem 3.5, é possível perceber que podemos colocar vários if-else aninhados. Ou seja, um if-else nos permite trabalhar em um primeiro intervalo. Após estar em um intervalo de interesse, pode-se testar outros. No caso anterior, a primeira decisão ocorre em números menores que 4 e o else (consequentemente números maiores ou iguais a 4). Dentro do intervalo dos números maiores que 4, testamos novamente no intervalo marcado pelo número 7.

3.3 Operador ternário

Agora suponha que tenhamos o seguinte problema.

■ **Examplo 3.3** Faça um programa que receba três números, mostre o maior e o menor. ■

Agora que você domina as expressões boolenas, ficou fácil resolver este problema. Uma possível solução é dada a seguir:

```
#include<stdio.h>
int main(){
  int a, b, c, maior, menor;
  puts("Entre com os números: ");
  scanf("%d %d %d", &a, &b, &c);
  if(a>=b && a >= c){//a é o maior
    maior = a;
    if(b<c){ //b é o menor
      menor = b;
    }else{//c então é o menor
      menor = c;
    }
  }
  if(b>=a && b >= c){//b é o maior
    maior = b;
    if(a<c){ //a é o menor
      menor = a;
    }else{//c então é o menor
      menor = c;
    }
  }
  if(c>=a && c >= b){//c é o maior
    maior = c;
    if(a<b){ //a é o menor
      menor = a;
    }else{//b então é o menor
      menor = b;
    }
  }
  printf("maior: %d menor %d \n", maior, menor);
  return 0;
}
```

O código acima vai funcionar e resolverá o problema. Entretanto, há outras maneiras utilizando-se de artefatos da própria linguagem. Um destes artefatos é o operador ternário. Permita-me fazer um código que resolve o mesmo problema anterior, porém usando o operador ternário.

```
#include<stdio.h>
int main(){
```

```
3   int a, b, c, maior, menor;
4   puts("Entre com os números: ");
5   scanf("%d %d %d", &a, &b, &c);
6   maior = a; menor = a;
7   //Leitura: se b for maior que o valor atual de maior, atualize o
        valor de maior com o valor de b. Em caso contrário, atribua o
        valor atual de maior.
8   maior = (b>maior)? b: maior;
9   maior = (c>maior)? c: maior;
10
11  //Leitura: se b for menor que o valor em menor, atribua o valor de
        b para a variável menor. Caso contrário, atribua o valor atual
        da variável menor.
12  menor = (b<menor)? b: menor;
13  menor = (c<menor)? c: menor;
14
15  printf("maior: %d menor %d \n", maior, menor);
16
17  return 0;
18 }
```

Considere o segundo exemplo a seguir:

■ **Examplo 3.4** Faça um programa que receba três números e mostre-os em ordem crescente.

```
1  #include<stdio.h>
2  int main() {
3    int a, b, c; // Entradas
4    int menor, mediano, maior;
5    puts("Entre com tres valores: ");
6    scanf("%d %d %d", &a, &b, &c);
7    if(a > b && a > c){
8      //a eh o maior
9      maior = a;
10     if(b>c){
11       menor = c;
12       mediano = b;
13     }else{
14       menor = b;
15       mediano = c;
16     }
17   }else{
18     if(b>a && b > c){
19       //b eh o maior
20       maior = b;
21       if(a>c){
22         menor = c;
```

```
23          mediano = a;
24        }else{
25          menor = a;
26          mediano=c;
27        }
28      }
29      else{
30        if(c>a && c >b){
31          //c eh o maior
32          maior = c;
33          if(a>b){
34            menor = b;
35            mediano = a;
36          }else{
37            menor = a;
38            mediano=b;
39          }
40        }
41      }
42    }
43    printf("%d %d %d\n", menor, mediano, maior );
44    return 0;
45  }
```

Uma outra solução seria:

```
1   #include<stdio.h>
2   int main() {
3     int a, b, c; // Entradas
4     int menor, mediano, maior;
5     puts("Entre com tres valores: ");
6     scanf("%d %d %d", &a, &b, &c);
7
8     if((a<b && a<c) && (b<c))
9       printf("%d %d %d",a, b, c);
10    if((a<b && a<c) && (c<b))
11      printf("%d %d %d",a, c, b);
12
13    if((b<a && b<c) && (a<c))
14      printf("%d %d %d",b, a, c);
15    if((b<a && b<c) && !(a<c))
16      printf("%d %d %d",b, c, a);
17
18    if((c<a && c<b) && (a<b))
19      printf("%d %d %d",c, a, b);
20    if((c<a && c<b) && !(a<b))
21      printf("%d %d %d",c, b, a);
```

```
22
23   return 0;
24 }
```

3.4 Switch

Switch é um seletor múltiplo fornecido pela linguagem C. Ele permite avaliar uma variável ou expressão que resulte em um inteiro.

```
1  switch(<expressao_que_resulte_inteiro>){
2    case valor1:
3      Bloco_para_valor1
4      break;
5    case valor2:
6      Bloco_para_valor2
7      break;
8    .
9    .
10   .
11   default:
12     Bloco_caso_nenhum_valor_esperado
13 }
```

Este comando é indicado para testar uma expressão que resulte em diversos valores pré-estabelecidos. Caso o break seja ocultado de algum bloco de comando, o valor continuará a ser executado.

```
1  #include<stdio.h>
2  int main(){
3    int op;
4    puts("Menu\n1->1 Para adicionar\n2-> Para remover\n0->Sair\nEntre
       com sua opcao:");
5    scanf("%d", &op);
6    switch(op){
7      case 1: puts("Adicionar hein!"); break;
8      case 2: puts("Remover mesmo?"); break;
9      case 0: puts("Bye bye!"); break;
10     default: puts("Você digitou errado!");
11   }
12   return 0;
13 }
```

```
1  #include<stdio.h>
2  #include<time.h>
3  /*Este programa toma como entrada uma data no formato dd/mm e
4  calcula quantos dias se passaram desde o inicio do ano corrente.
5  */
```

```
6  unsigned int getCurrentYear(){
7    time_t timeval;
8    struct tm *tp;
9    time (&timeval);
10   tp = gmtime(&timeval);
11   return (unsigned) tp->tm_year+1900;
12 }
13
14 unsigned int isBisexto(unsigned int a){
15   return (a % 4 == 0) && ((a%400==0)||(a % 100 != 0));
16 }
17
18 int main(){
19   unsigned int mm, dd, yy = getCurrentYear();
20   printf("\n current year: %u\n",yy);
21   printf("\n Insira uma data no formato dd/mm : ");
22   if(scanf("%u/%u", &dd, &mm) != 2)
23   {
24     printf("\nOs dados nao foram lidos corretamente!\n");
25     return 0;
26   }
27   char dateOk = 1;
28   unsigned int qtdDias = dd;
29   switch(mm){
30     case 1:
31     case 3:
32     case 5:
33     case 7:
34     case 8:
35     case 10:
36     case 12: dateOk = dd<=31; break;
37     case 4:
38     case 6:
39     case 9:
40     case 11: dateOk = dd<=30; break;
41     case 2: dateOk = ((isBisexto(yy) && dd == 29) || dd<29); break;
42     default: dateOk = 0;
43   }
44   if(dateOk){
45     switch(mm-1){
46       case 11: qtdDias +=30;
47       case 10: qtdDias+=31;
48       case 9: qtdDias +=30;
49       case 8: qtdDias+=31;
50       case 7: qtdDias +=31;
51       case 6: qtdDias+=30;
```

```
52        case 5: qtdDias +=31;
53        case 4: qtdDias+=30;
54        case 3: qtdDias+=31;
55        case 2: qtdDias+=28+isBisexto(yy);
56        case 1: qtdDias+=31;
57      }
58
59      printf("\nDo inicio do ano, se passaram %u dias!\n", qtdDias);
60    }else
61      printf("\nInsira uma data valida!\n");
62
63    return 0;
64  }
```

Lista 3.2 - Exercícios de fixação

Problema 3.1 Faça um algoritmo que receba um número e mostre a mensagem ?maior que dez!? caso este número seja maior que 10.

Problema 3.2 Faça um programa que receba três números garantidamente em ordem crescente (não precisa testar). Leia agora um quarto número. Assumindo que os quatro números lidos são diferentes uns dos outros, mostre os quatro números em ordem crescente.

Problema 3.3 Escrever um algoritmo que leia o nome e as três notas obtidas por um aluno durante o semestre. Calcular a sua média (aritmética), informar o nome e sua situação a saber: aprovado (media >= 7) reprovado (media < 4) e recuperação (média entre 4.0 a 6.999).

Problema 3.4 Faça um algoritmo que receba um número e diga se este número está no intervalo entre 100 e 200.

Problema 3.5 Escreva um algoritmo para ler o nome de 2 times e o número de gols marcados. Escrever o nome do vencedor. Caso não haja vencedor deverá ser impresso a palavra EMPATE.

Problema 3.6 A jornada de trabalho semanal é de 40 horas. O funcionário que trabalhar mais de 40 horas receberá hora extra, cujo valor é o valor da hora regular com um acréscimo de 50%. Escreva um algoritmo que leia o número de horas trabalhadas em um mês, o salário por hora e escreva o salário total do funcionário (considere que o mês possua 4 semanas exatas).

Problema 3.7 Escreva um algoritmo para ler as 3 notas obtidas por um aluno nas 3 verificações e a média dos exercícios que fazem parte da avaliação. Calcular a média de aproveitamento, usando a fórmula: (N1 + N2 * 2 + N3 * 3 + Média dos exercícios Média de aproveitamento)/7. A atribuição dos conceitos obedece a tabela abaixo:

Média de aproveitamento	Conceito
> = 9,0	A
> = 7,5 e < 9,0	B
> = 6,0 e < 7,5	C
<6	D

Problema 3.8 Uma empresa dará aumento aos seus funcionários, de acordo com sua Classe: a) Classe A = 0,10(10%) de aumento; b) Classe B = 0,15 (15%) de aumento; c) Classe C = 0,20(20%) de aumento. Faça um programa que leia o salário e a classe do funcionário, calcule e exiba os salários com os devidos aumentos.

Problema 3.9 Faça um algoritmo que receba o número do mês e mostre o nome do mês correspondente. Valide se é um mês válido.

Problema 3.10 Dados três valores A, B e C, em que A e B são números reais e C é um caractere, pede-se para imprimir o resultado da operação de A por B se C for um símbolo de operador aritmético; caso contrário deve ser impressa uma mensagem de operador não definido. Obs.: Tratar erro de divisão por zero.

3.5 Gerando números aleatórios

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<time.h>
4
5  int main(){
6    srand(time(NULL));
7    int a;
8    a = rand()%100;
9    printf("Num gerado: %d \n", a);
10
11   return 0;
12 }
```

Jogo Hi-lo

O jogo Hi-Lo de uma chance consiste em gerar um número aleatório em um determinado intervalo, solicitar o palpite do usuário e emitir o seguinte parecer:

1. Caso o palpite seja igual ao número gerado, então imprima a mensagem que a pessoa acertou.
2. Caso o palpite seja maior do que o número gerado, informe a pessoa desta informação;
3. Caso o palpite seja menor do que o número gerado, informe a pessoa desta informação.

Faça o código do jogo.

Agora imagine que se deseja dar pelo menos 3 chances à pessoa do jogo. Logo teríamos que copiar e colar o trecho de código.

Podemos contornar isso com um laço de repetição.

Bom, agora vamos melhorar este negócio e criar o primeiro jogo para chamar os amigos para participar. Tudo com dinheirinho de mentira, ok? Se atente às seguintes regras:

1. Gerar o número aleatório no intervalo de 0 a 99;
2. Ler o investimento na máquina (aceitar apenas notas de 10, 20, 50 e 100);
3. Acrescente 5 reais ao investimento
4. Loop principal do jogo (enquanto houver crédito)
   (a) Exibir os créditos
   (b) Ler o palpite e julgar:
      i. Acertou: devolva os créditos ao usuário
      ii. Errou:
         - dê a dica (se o palpite foi maior ou menor)
         - desconte 0,50 dos créditos (cada palpite errado custa 50 centavos);

Funcionaria com um cientista da computação?

Imagine a utilização da seguinte estratégia:

1. Palpite sempre no meio do intervalo conhecido.
2. Caso o papite seja maior que o valor oculto, descarte a metade anterior e procure apenas na metade onde se sabe que o número se encontra.

Por exemplo, suponha que o número secreto seja 73. Se eu fizer o palpite em 50 (que é aproximadamente o valor do meio entre 0 e 99), o programa me informará que meu palpite foi menor do que o número oculto. Certo. Então isso significa que meu espaço de busca diminuiu pela metade (eu não vou tentar chutar um número menor que 50.)!

Na próxima rodada, eu mando 74. E o programa vai me dizer que o palpite está acima. Ok. Então eu concluo que o número está entre 51 e 73. Logo, eu palpito 62. O programa vai me dizer que meu palpite está menor. Eu concluo que o número está entre 63 e 73. Logo meu palpite é 68. E o programa me diz que o meu palpite é menor. Logo eu coloco meu próximo palpite em 71. E o programa me informa novamente que o palpite é menor. Logo eu concluo, está entre 72 e 73. Com no máximo 2 palpites eu acerto o número. Perceba que eu gastei no máximo 7 rodadas. Isto quer dizer que se cada palpite errôneo custa R$0,5, isso significa que no pior dos casos, o cientista da computação ganhará R$1,50 toda vez que jogar!

Na verdade esse método é o utilizado na busca binária em vetores ordenados, que veremos em algoritmos para vetores. A explicação matemática para o que ocorreu anteriormente é que cada vez que o usuário palpita daquela forma, ele divide o espaço de busca em dois. Logo, podemos concluir que o número máximo de jogadas seria $\lceil \log_2(100) \rceil$, considerando que os números serão gerados entre 0 e 99. Mas eu realmente duvido que o seu tio lá do "é pra ver ou pra comer" vai conseguir chegar nesta estratégia (ao menos que ele seja um cientista da computação).