Jogo da adivinhação

Parte 3

Execução repetida de trechos do programa

Se nosso jogo permitir que o usuário tenha três chances, teríamos que repetir o código 3 vezes

Isso seria ineficiente e altamente sujeito a erros

Dar manutenção neste programa significaria repetir correções e alterações em diversas linhas

Imagine então se o usuário tivesse infinitas chances… como resolver esta questão?

Execução repetida de trechos do programa

Para executar repetidas vezes trechos de código, utilizamos estruturas de repetição, também conhecidas como loops

Vamos ver dois tipos de loops: for e while

Usamos for quando sabemos exatamente quantas vezes o trecho de código deve se repetir

Quando a quantidade de vezes pode variar, ou até ser infinita, recomendamos o uso do while

Estrutura de repetição FOR

Uma linha de comando for possui três partes:

- Declaração e inicialização da variável que será utilizada como contador
- Condição de parada do loop
- Incremento ou decremento do loop

Nosso for ficaria assim:

```
for(int tentativas=0;tentativas<3;tentativas++)
```

Todo o trecho a ser repetido ficaria dentro das chaves que se seguiriam ao for

Não esqueça de identar o que estiver dentro destas chaves

Atividades

1 - Caso o usuário acerte o número, não precisamos continuar dentro do for. Como resolver isso?

2 - Imprima o número da tentativa do jogador. Exemplo: “Tentativa 1”, “Tentativa 2”, “Tentativa 3”. Obs: use o contador do for.

3 - Pesquise o termo escopo de variáveis. Qual o escopo da variável tentativas utilizada no exemplo do slide anterior?

4 - Se quiséssemos que o jogador possa ter infinitas tentativas, deveríamos usar for ou while?

Usando Constantes

Alguns valores chaves de um programa devem ser definidos de forma “centralizada” e não em qualquer parte do código

Isto evita erros e facilita a manutenção do programa

Em nosso programa, podemos definir no início do código o número de tentativas

Como este valor não vai ser alterado ao longo do código, podemos declará-lo como uma constante, não como uma variável

```
#define NUMERO_MAX_TENTATIVAS 3
```

```
for(int tentativas=0;tentativas<NUMERO_MAX_TENTATIVAS;tentativas++)
```

Usando Constantes

As constantes, como o próprio nome diz, não podem ser alteradas durante a execução do programa

Convém nomear as constantes de forma diferente das variáveis, para facilitar a leitura do código

Por isso foi utilizado maiúsculas com underscore

Usando else if

Caso se queira evitar estruturas condicionais aninhadas, podemos promover uma alteração

Ao utilizar else if, podemos testar novas condições a cada else

```
if(palpite==numeroSecreto)
{
    printf("\n Parabéns, você acertou. O número secreto era %d \n", numeroSecreto);
    printf("\n Jogue outra partida");
    //sai do loop
    break;
}
//verifica se palpite é maior que o númeroSecreto
else if(palpite>numeroSecreto)
{
    printf("\n O número secreto é menor que %d \n", palpite);
    printf("\n Tente novamente \n");
}
//verifica se palpite é menor que o númeroSecreto
else if(palpite<numeroSecreto)
{
    printf("\n O número secreto é maior que %d \n", palpite);
    printf("\n Tente novamente \n");
}
```

Ignorando números negativos (opcional)

Tratar valores inválidos faz parte da maioria dos softwares

Para tanto, vamos incluir um if após a leitura do palpite e testar se ele é negativo

Se for, vamos ignorar a tentativa, voltando ao início do loop

```
if(palpite<0)
{
    printf("\n Digite apenas números maiores que zero");
    //decrementa a variável tentativas para que
    //esta tentativa não conte
    tentativas--;
    //ignora o restante do código e volta ao início do loop
    continue;
}
```

Acrescentando pontuação do jogador

Nossa regra é a seguinte: o jogador começa com 1000 pontos

A cada rodada subtraímos da pontuação a metade da diferença entre o número secreto e o chute

Exemplo: Número secreto=30, Chute=20, pontos=pontos-(30-20)/2

Vamos declarar os pontos como uma variável real (pois ela pode conter casas decimais)

```
float pontos=1000;
```

Acrescentando pontuação do jogador

Quando houver um erro, fazemos a operação matemática

```
pontos=pontos-fabs(numeroSecreto-palpite)/2;
```

fabs( ) é uma função que retorna o valor absoluto, não interessando se número é positivo ou negativo

Para usar esta função, devemos incluir também a biblioteca de funções matemáticas

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

Acrescentando pontuação do jogador

Por fim, imprimimos a pontuação antes de encerrar o jogo

```
printf("\n Sua pontuação foi %.2f",pontos);
printf("\n**********************************\n");
printf("Fim da partida \n");
printf("**********************************\n");
```

%.2f significa que iremos imprimir uma variável numérica com ponto flutuante com duas casas decimais

Atividade

1 - Pesquise os operadores matemáticos da linguagem C.

2 - Pesquise os tipos de dados e suas respectivas capacidades de armazenamento

3 - Em uma operação matemática, como converter os números de inteiro para ponto flutuante e vice-versa? Isto pode gerar perda de valores?

Variando o número secreto

Até aqui o número secreto foi pré-definido no código do jogo

Se quiséssemos alterá-lo, teríamos que alterar o código, compilar e disponibilizar um novo executável para os jogadores. Algo pouco prático

Usando números randômicos (ou aleatórios), nosso jogo gerará um número diferente a cada nova execução do jogo

Podemos usar a função rand( ) da biblioteca stdlib.h

Experimente gerar e imprimir números aleatórios

```
int aleatorio1, aleatorio2;

aleatorio1=rand();
aleatorio2=rand();

printf("\n Número aleatório: %d",aleatorio1);
printf("\n Número aleatório: %d",aleatorio2);
```

Variando o número secreto

Se você rodar este programa várias vezes, verá que os mesmos números foram gerados

Então podemos dizer que são números pseudoaleatórios

Para evitar isso usaremos uma “semente”, que seria uma variável fornecida a essa função que garantiria de fato números diferentes

Se repetíssemos sempre a mesma semente, continuaríamos tendo os mesmos números

Então utilizaremos como semente a data completa atual (anos, mês, dia, hora, minutos e segundos), algo que nunca se repete

Variando o número secreto

A função time( ), que retorna o número de segundos passados desde a data conhecida como Unix Epoch

```
int aleatorio1, aleatorio2;
//obtém número de segundos desde UNIX EPOCH
int segundos=time(NULL);
//função que define a semente com base no número passado como parâmetro
srand(segundos);
aleatorio1=rand();
aleatorio2=rand();

printf("\n Número aleatório: %d",aleatorio1);
printf("\n Número aleatório: %d",aleatorio2);
```

Variando o número secreto

Ainda precisamos de uma última alteração para utilizar este código em nosso jogo

O número gerado é muito grande, portanto precisamos definir um intervalo

Se calcularmos o resto do número gerado por 100, teremos um intervalo de 0 a 99

Basta fazer a alteração abaixo

```
aleatorio1=rand()%100;
aleatorio2=rand()%100;
```

Variando o número secreto

Voltando ao jogo, definiremos assim o número secreto

```
//obtém número de segundos desde UNIX EPOCH
int segundos=time(NULL);
//função que define a semente com base no número passado como parâmetro
srand(segundos);
numeroSecreto=rand()%100;
```

Seria interessante imprimirmos o número secreto quando encerrarmos o jogo

Código completo

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

#define NUMERO_MAX_TENTATIVAS 3

int main(int argc, char **argv)
{
    //variável que armazenará número a ser advinhado
    int numeroSecreto, palpite;
    float pontos=1000;
    //obtém número de segundos desde UNIX EPOCH
    int segundos=time(NULL);
    //função que define a semente com base no número passado como parâmetro
    srand(segundos);
    numeroSecreto=rand()%100;

    printf("*********************************\n");
    printf("Bem-vindo ao Jogo de Advinhação \n");
    printf("*********************************\n");
    printf(":)\n");
```

```c
for(int tentativas=0;tentativas<NUMERO_MAX_TENTATIVAS;tentativas++)
{
    printf("\n Esta é sua tentativa número %d",tentativas+1);
    printf("\n Digite um número:");
    scanf("%d",&palpite);

    if(palpite<0)
    {
        printf("\n Digite apenas números positivos");

        //decrementa a variável tentativas para que
        //esta tentativa não conte
        tentativas--;
        //ignora o restante do código e volta ao início do loop
        continue;
    }
    else if(palpite==numeroSecreto)
    {
        printf("\n Parabéns, você acertou. O número secreto era %d \n", numeroSecreto);
        printf("\n Jogue outra partida");
        //sai do loop
        break;
    }
    //verifica se palpite é maior que o númeroSecreto
    else if(palpite>numeroSecreto)
    {
        printf("\n O número secreto é menor que %d \n", palpite);

    }
```

```c
    //verifica se palpite é menor que o númeroSecreto
    else if(palpite<numeroSecreto)
    {
        printf("\n O número secreto é maior que %d \n", palpite);

    }
    pontos=pontos-fabs(numeroSecreto-palpite)/2;

}

printf("\n O número secreto era: %d",numeroSecreto);
printf("\n Sua pontuação foi %.2f",pontos);
printf("\n**********************************\n");
printf("Fim da partida \n");
printf("**********************************\n");

return 0;
}
```