Departamento de Ciência da Computação - IME - USP

MAC2166 – Introdução à Ciência da Computação

Escola Politécnica - Computação / Elétrica - Primeiro Semestre de 2022

Exercício-Programa 1 (EP1)
Data de Entrega: **30 de abril**

Para se preparar bem para o desenvolvimento de seu EP1, cuja descrição se inicia na próxima página, leia com atenção as instruções abaixo.

- Utilize **somente** os recursos da linguagem que aprendeu nas aulas.
- Veja em `https://www.ime.usp.br/~mac2166/infoepsC/` as instruções de entrega dos exercícios-programa e atente para as instruções de preenchimento do cabeçalho do seu programa.
- Leia um FAQ sobre compilação em `https://www.ime.usp.br/~mac2166/compilacao/`.
- Sempre compile seus programas com as opções **-Wall -ansi -pedantic -O2**.
- **Importante**: os casos de teste disponíveis na correção automática do e-disciplinas[1] servem somente para ajudar na análise de seu programa, não tendo nenhuma influência na nota do EP. Seu programa deve:
  - funcionar para qualquer entrada possível;
  - estar em conformidade com o enunciado;
  - estar bem estruturado;
  - ser de fácil compreensão, com o uso padronizado da linguagem C.

[1] veja o item "EP1 - Entrega" no e-disciplinas

EP1: NUMLE, um Wordle numérico - Versão final

Neste exercício-programa você implementará a versão final do Numle, cuja variante mais simples foi o assunto do EP0. A nota deste exercício-programa é de 9.0 (nove) pontos, que será somada à nota do EP0 para, assim, compor a nota final do EP1.

Numle

O Wordle é um jogo de palavras que ficou na moda nos últimos tempos. Ele foi comprado pelo jornal NY Times, e tem versões disponíveis em várias línguas do mundo.

Já trabalhamos um pouco com uma variante numérica do Wordle, chamado de **Numle**, em que o jogador realiza algumas tentativas para adivinhar um número formado por um único dígito. Nosso objetivo agora é tornar o jogo ainda mais interessante! Você deve escrever um programa em C no qual o objetivo do usuário é acertar um número inteiro positivo com entre 0 e 99999 gerado aleatoriamente pelo programa.

Inicialmente seu programa pede ao usuário uma semente (um inteiro entre 0 e 10000) que será usada para sortear um número aleatório entre 0 e 99999, que chamamos de `senha`. Depois o programa deve perguntar ao usuário quantas tentativas (um inteiro entre 1 e 10) ele gostaria de ter para adivinhar a senha.

Consideramos que a senha tem sempre 5 dígitos. Por exemplo, para nosso jogo, os números 30 e 1 são vistos, respectivamente, como 00030 e 00001. No que segue, chamamos de `tentativa` o número de 5 dígitos digitado pelo usuário do programa. Vamos às regras do jogo!

- Os dígitos da `tentativa` são analisados da esquerda para a direita, ou seja, do mais significativo para o menos significativo e as informações (descritas a seguir) dadas pelo programa sobre dígitos nas posições corretas e dígitos em posições incorretas devem ser dadas sempre da esquerda para a direita.
- Para cada dígito de `tentativa` que está na posição correta, seu programa deve dar essa informação ao usuário. Veja abaixo alguns exemplos:
  - se `senha` = 12313 e `tentativa` = 11511, seu programa deve indicar que o primeiro e o quarto dígitos estão corretos. Para os demais dígitos, o programa não deve indicar nada. Neste caso, dizemos que o 1o e o 4o dígitos de `senha` foram *usados* (isso será relevante a seguir).
  - se `senha` = 12345 e `tentativa` = 54123, nenhum dígito de `tentativa` está na posição correta. Neste caso, todos os dígitos de `senha` foram usados.
- Uma vez que os dígitos em posição correta de `tentativa` forem identificados, verificamos quais dígitos de `tentativa` aparecem em posição incorreta (note que, ao identificarmos os dígitos de `tentativa` em posição correta, usamos alguns dígitos de `senha`). Os dígitos de `tentativa` que ocorrem em posição incorreta em `senha` devem ser informados ao usuário. Por exemplo, no exemplo acima, em que `senha` = 12345 e `tentativa` = 54123, o programa deverá informar que todos os dígitos aparecem em posição incorreta. Porém, um mesmo dígito de `senha` ou de `tentativa` **não** poderá ser usado mais de uma vez para dar informações ao usuário. Veja abaixo alguns exemplos:

- Se `senha = 22221` e `tentativa` = 11110, seu programa deverá indicar que o primeiro dígito da tentativa aparece em posição incorreta. Neste processo, o 5o dígito de `senha` é usado. Portanto, não falamos nada sobre o 2o, 3o e 4o dígitos de `tentativa`.
- Se `senha` = 22211 e `tentativa` = 11100, seu programa deverá indicar que o primeiro e o segundo dígitos aparecem em posição incorreta, pois há exatamente dois dígitos iguais a 1 em `senha` (e eles foram usados para dizer que o 1o e 2o dígito de `tentativa` aparecem em posição incorreta).
- Se `senha` = 22221 e `tentativa` = 11111 seu programa deverá **apenas** indicar que o quinto dígito ocorre na posição correta, uma vez que só há um dígito igual a 1 em `senha`.
- Se `senha` = 12123 e `tentativa` = 14211, seu programa deve indicar que o 1o dígito da tentativa ocorre na posição correta, de modo que o 1o dígito de `senha` é usado. Seu programa deve indicar também que o terceiro dígito está em posição incorreta, pois há digitos iguais a 2 que não foram usados em `senha`. Por fim, seu programa deve indicar que o 4o dígito está em posição incorreta. Note que nada deve ser dito sobre o 5o dígito de `tentativa`, dado que os dois dígitos 1 de `senha` já foram usados.

Caso você ainda tenha dúvidas sobre as regras para fornecer informações sobre dígitos em posições incorretas, a seguir temos uma explicação precisa: para cada dígito d que ocorre em `tentativa`, seja x_{senha} a quantidade de vezes que d ocorre em `senha`. Ademais, sejam x_{certa} e x_{errada} a quantidade de vezes que d ocorre, respectivamente, na posição correta e na posição errada em `tentativa`. Seu programa deve indicar, para os primeiros $\min\{x_{\text{senha}} - x_{\text{certa}}, x_{\text{errada}}\}$ dígitos em que d aparece, que eles estão na posição errada.

O usuário fará chutes para adivinhar a senha sorteada. Se conseguir acertar, o programa deverá indicar isso. Caso contrário, se as tentativas do usuário se esgotarem, o programa encerra e indica que o usuário perdeu.

Exemplos de execuções do programa

```
Bem vinda(o) ao Numle
Digite a semente para sortear a senha (0 a 10000): 1234
Quantidade de tentativas (1 a 10): 6

Digite a tentativa (0 a 99999): 01234
Segundo digito em posicao incorreta.

Digite a tentativa (0 a 99999): 56789
Quarto digito certo!
Quinto digito em posicao incorreta.

Digite a tentativa (0 a 99999): 99181
Primeiro digito certo!
Segundo digito certo!
Quarto digito certo!
Quinto digito certo!
```

```
Digite a tentativa (0 a 99999): 99981
Voce acertou! A senha eh de fato 99981.
```

```
Bem vinda(o) ao Numle
Digite a semente para sortear a senha (0 a 10000): 7490
Quantidade de tentativas (1 a 10): 6

Digite a tentativa (0 a 99999): 13579
Terceiro digito certo!
Quinto digito certo!

Digite a tentativa (0 a 99999): 02468
Primeiro digito em posicao incorreta.
Quinto digito em posicao incorreta.

Digite a tentativa (0 a 99999): 80559
Primeiro digito certo!
Segundo digito certo!
Terceiro digito certo!
Quinto digito certo!

Digite a tentativa (0 a 99999): 80589
Voce acertou! A senha eh de fato 80589.
```

```
Bem vinda(o) ao Numle
Digite a semente para sortear a senha (0 a 10000): 639
Quantidade de tentativas (1 a 10): 3

Digite a tentativa (0 a 99999): 12145
Segundo digito em posicao incorreta.
Quarto digito em posicao incorreta.

Digite a tentativa (0 a 99999): 21414
Primeiro digito em posicao incorreta.
Terceiro digito certo!

Digite a tentativa (0 a 99999): 13476
Terceiro digito certo!
Voce perdeu! A senha era 8402.
```

Você deve seguir **obrigatoriamente** o formato acima.

Como gerar números aleatórios

O seguinte trecho inicial de código deve ser utilizado como base para seu EP1 (Perceba que há uma leve mudança no código abaixo se comparado com o código disponibilizado no EP0).

```
#include <stdio.h>

int main() {
  int semente, senha;

  printf("Bem vinda(o) ao Numle\n");
  printf("Digite a semente para sortear a senha (0 a 10000): ");
  scanf("%d", &semente);
  semente = semente % 134456;

  /* sorteia um numero 'aleatorio' entre 0 e 99999 */
  senha = ((8121 * semente + 28411) % 134456) % 100000;
```

Desejamos um bom trabalho a todos!