Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação

Trabalho Final – Jogo Senha
Disciplina Programação (CK0110) – Semestre 2014.2

Prof. Miguel Franklin

***** PARA EQUIPES DE NO MÁXIMO 2 (DOIS) ALUNOS *****

Desenvolver um programa em linguagem C representando o jogo Senha, onde o usuário deverá ser capaz de jogar contra o computador.

Requisitos Funcionais:

1. O usuário deverá, no início do jogo, definir os seguintes parâmetros da partida:
    a. *N*: Tamanho da senha (de 4 a 12)
    b. *PC*: Possibilidades de cores (de 3 a 6)
    c. *Lim*: Limite de tentativas de adivinhação (a partir de 1, sem limite superior)
2. O usuário deverá, a cada rodada, entrar uma tentativa de adivinhação, que é composta por uma sequência de códigos de cores, usando-se as possibilidades de cores a seguir:
    a. Vermelho – vm
    b. Verde – vd
    c. Amarelo – am
    d. Azul – az
    e. Magenta – mg
    f. Ciano – ci
3. Antes de solicitar a tentativa do jogador, o computador deve informar as cores que estão disponíveis, dentre as possibilidades citadas acima. Caso o jogo esteja configurado para utilizar menos do que 6 cores, as cores a serem escolhidas devem seguir a ordem de prioridade listada acima. Por exemplo, se for escolhido ter apenas 3 possibilidades, estas serão, necessariamente: “vm”, “vd” e “am”.
4. Ao solicitar uma tentativa de adivinhação, o usuário deve entrar uma sequência de *N* cores, separadas por vírgula. O computador deve ser capaz de compreender sequências que tenham espaços depois das vírgulas (Ex. “vd, vm, am” no lugar de “vd,vm,am”).
5. O computador deve validar a entrada do usuário, considerando a diversidade e a quantidade de cores entradas. Caso o usuário já tenha tentado a combinação entrada, considerando o histórico de todas as combinações já entradas, o sistema deverá exibir a mensagem “Você já tentou esta combinação antes”, ignorando-a e solicitando uma nova combinação. **Uma senha pode ter elementos de cores repetidas.**
6. Caso a combinação entrada pelo usuário seja inédita, o computador deverá analisá-la e mostrar a combinação em cores e o resultado da análise ao lado direito, indicando cada acerto de cor fora da posição com um símbolo ‘!’ e o acerto de uma cor na posição correta com um símbolo ‘@’. O programa não deverá indicar em que posições houve acerto. Considere o exemplo a seguir:

Exemplo: considere um jogo cujas variáveis são:

- *N* = 4
- *PC* = 6
- *Lim* = 100

Observação: O que for indicado em negrito é o que o computador apresenta ao usuário. O que for apresentado em itálico é o que o usuário entrou.

Consideremos que a senha sorteada pelo computador seja: “am, az, mg, vm”

```
Entre com uma combinação de 4 elementos dentre as 5 cores seguintes:
- vm: vermelho
- vd: verde
- am: amarelo
- az: azul
- mg: magenta

Entre a sua tentativa, separando os elementos por vírgula:
vd,am,az,vm

o o o o  ->  ! ! @

Esta foi a tentativa de número 1. Você não acertou. Entre sua nova tentativa:
```

Obs. (1): Observe que o primeiro símbolo ‘!’ diz respeito ao fato da senha conter “am”, mas que tal cor não está no local correto. O segundo símbolo ‘!’ diz respeito ao fato da senha conter “az”, mas que tal cor não está no local correto. O símbolo ‘@’ diz respeito ao fato da senha conter “vm” exatamente no local que o usuário indicou (posição 4).

Obs. (2) A representação em cores das jogadas deve ser mostradas ao usuário. Utilize no seu projeto o arquivo cores.h disponibilizado no SIGAA para este fim.

7. Se o usuário acertar a senha antes do limite de jogadas, o computador deve exibir a mensagem: “Parabéns. Você conseguiu adivinhar a senha com *M* tentativas.”, onde *M* foi a quantidade de tentativas do usuário, não considerando aquelas eventualmente repetidas.

Requisitos Não Funcionais:

1. O programa deverá utilizar o mínimo de memória possível.
2. O programa deverá conter a implementação de **pelo menos** uma estrutura de dados utilizando registros e gerenciamento dinâmico de memória.
3. O projeto deverá ser constituído por códigos fontes separados, de acordo com características funcionais (Ex. estrutura de dados, programa principal, funções auxiliares, etc.). Todos os arquivos-fonte com o respectivo Makefile para construir o projeto devem ser disponibilizados na entrega.
4. O dimensionamento das variáveis a serem utilizadas deve otimizar a ocupação de memória. (Ex. se uma jogada pode ter de 4 a 12 cores, não convém estipular um vetor de 12 posições para representar uma jogada.

Critérios de Avaliação

A avaliação será realizada em duas fases:

1. Análise do código-fonte;
2. Análise da execução do programa (teste);
3. Apresentação do protótipo em laboratório (se solicitado pelo professor).

O código-fonte será avaliado de acordo com os seguintes critérios qualitativos:

i. Eficácia do programa em suprir todos os requisitos funcionais e não funcionais;
ii. Eficiência do programa (otimização);
iii. Organização do código (uso racional de subprogramas, estruturas, etc.);
iv. Legibilidade do código (uso de endentação e semântica dos identificadores de variáveis);
v. Documentação (comentários dentro do código fonte).

Obviamente, funcionalidades adicionais às que foram solicitadas neste documento são bem vindas e serão gratificadas na nota (na medida do possível). O código-fonte deve conter, em comentário no início, os nomes e matrículas dos alunos que compõem o grupo. O código-fonte deve ser submetido na data fixada através de servidor de *upload*, a ser definido. A apresentação do protótipo será marcada em seguida.

Lembramos que todos os programas serão submetidos a análise léxica automática, que pode evidenciar cópia de código.

Os trabalhos serão corrigidos no **Linux**. Portanto, certifique-se que o trabalho feito no Windows também compila e roda no Linux.

Prazo de Entrega: 20 de novembro de 2014, até 23:59.

Upload através do SIGAA (Turma Virtual)

Não serão aceitas entregas por e-mail (a não ser que haja algum problema com a submissão pelo *SIGAA*).