

As questões e exercícios de programação a seguir foram retiradas do livro “*Programming Abstractions in C: A Second Course in Computer Science*”, de Eric S. Roberts (Addison-Wesley, 1998), Capítulo 3: *Libraries and Interfaces*.

1 Questões de Revisão

Responda às questões abaixo, de forma **manuscrita**, em papel almaço. Entregue diretamente ao professor na data indicada.

- (a) Defina os seguintes termos: interface, pacote, abstração, implementação e cliente.
- (b) Em suas próprias palavras, descreva a diferença de perspectiva entre um programador que escreva a implementação de uma biblioteca e um programador que escreve o programa cliente dessa biblioteca.
- (c) Como as interfaces são representadas em C?
- (d) Cite e explique 5 critérios para o bom projeto de interfaces.
- (e) Por que é importante que uma interface seja estável?
- (f) O que significa estender uma interface?
- (g) Por que os comentários são extremamente importantes em uma interface?
- (h) Se você estiver criando uma interface chamada de “`magica.h`”, qual seria o *boilerplate* dessa interface? Para que serve?
- (i) Por que os valores gerados pela função “`rand`” são chamados de pseudo-aleatórios?
- (j) Você poderia usar a seguinte atribuição múltipla para simular o processo de jogar um dado duas vezes? Por quê?

```
d1 = d2 = inteiro_aleatorio(1, 6);
```

- (k) Explique o funcionamento dos quatro passos necessários para converter o resultado da função “`rand`” em um valor inteiro dentro dos limites dados pelos argumentos `min` e `max`.
- (l) Que idioma você utilizaria para processar os caracteres de uma string que você entende como um array de caracteres? Como esse idioma seria alterado se você entender essa string como um ponteiro para um caractere?
- (m) Qual a principal diferença entre os modelos de abstração fornecidos pelas interfaces `string.h`, `strlib.h` e `CRpaic.h`?

- (n) Assumindo que `s1` e `s2` sejam strings, descreva o efeito da expressão condicional no seguinte trecho de código:

```
if (s1 == s2)
```

- (o) O que é o *buffer overflow*?
- (p) Qual o propósito do tipo “`FILE *`”? Compreender a estrutura subjacente desse tipo é importante para a maioria dos programadores?
- (q) O que significa a frase “abrir um arquivo”?
- (r) O segundo argumento da função “`fopen`” é geralmente umas das seguintes strings: “`r`”, “`w`” ou “`a`”. Qual o significado desses argumentos e o que cada um deles faz?
- (s) Como a função “`fopen`” retorna uma falha para seu chamador?
- (t) A interface “`stdio.h`” automaticamente define três arquivos padrão (*standard files*). Quais são seus nomes? Para que cada um deles serve?
- (u) Se você estiver usando a função “`getc`”, como você detectará o final de um arquivo?
- (v) Para que serve a função “`ungetc`”?
- (w) Que passos são necessários no processo de atualização de um arquivo?
- (x) O que significa usar um asterisco (*) como o campo comprimento (*width*) ou precisão (*precision*) em um especificador de formato da função “`printf`”?
- (y) Você concorda com a seguinte frase (explique o motivo): “Com o passar dos anos a função `scanf` se mostrou uma funcionalidade extremamente útil da Biblioteca C Padrão”.
- (z) Quando você estiver utilizando as funções trigonométricas da interface “`math.h`”, como você converteria um ângulo de graus para radianos?

2 Exercícios de Programação

Em cada exercício você deve criar interfaces apropriadas (*header files*), implementar essas interfaces e criar um programa cliente que utiliza essas interfaces. Entregue no Autolab um arquivo compactado no formato ZIP contendo a interface, a implementação da interface e o programa cliente. Atenção: compacte diretamente os arquivos, não faça a compactação de um diretório contendo os arquivos!

- (a) Escreva um programa que repetidamente cria um número aleatório entre 0 e 1 e mostre a média desses números após uma certa quantidade de rodadas solicitadas pelo usuário. Por exemplo:

```
Este programa calcula a média de uma série de números aleatórios
entre 0 e 1.
Quantas rodadas serão executadas? 10000
A média após 10000 rodadas é 0.501493
```

Se o gerador de números aleatórios estiver funcionando corretamente, a média deve se tornar mais e mais perto de 0,5 à medida que o número de rodadas aumenta.

- (b) Escreva um programa que simula jogar uma moeda repetidamente para obter “cara” ou “coroa”, até que 3 caras consecutivas sejam obtidas. Nesse ponto seu programa deve exibir o número total de jogadas que foram feitas. Por exemplo:

```

coroa
cara
cara
coroa
coroa
cara
coroa
cara
cara
cara
Foram necessárias 10 jogadas para obter 3 caras consecutivas.

```

- (c) Em cassinos como os de Monte Carlo ou Las Vegas, uma das máquinas de aposta mais famosas é o caça-níquel — o “bandido de um braço”. Uma máquina caça-níquel típica tem três rodas que giram atrás de uma janela. Cada roda é marcada com os seguintes símbolos: CHERRY, LEMON, ORANGE, PLUM, BELL e BAR. A janela permite que você veja apenas um único símbolo em cada uma das três rodas. A janela, por exemplo, pode mostrar a seguinte posição das rodas:



Se você colocar um dólar na máquina e puxar a alavanca lateral, as rodas começam a girar e eventualmente param em alguma configuração. Se a configuração obtida bater com uma das configurações vencedoras, você ganha o valor daquela configuração (existem sete configurações vencedoras e cada uma tem um prêmio diferente). Se a configuração obtida não bater com nenhuma configuração vencedora, você perde um dólar. A seguinte tabela mostra as configurações vencedoras e os prêmios de cada uma:

BAR	BAR	BAR	250
BELL	BELL	BELL/BAR	20
PLUM	PLUM	PLUM/BAR	14
ORANGE	ORANGE	ORANGE/BAR	10
CHERRY	CHERRY	CHERRY	7
CHERRY	CHERRY	—	5
CHERRY	—	—	2
LEMON	—	—	0
—	LEMON	—	0
—	—	LEMON	0

A notação “BELL/BAR” significa que a configuração é considerada vencedora se aparecer um BELL ou um BARR naquela roda, e a notação “—” significa que qualquer símbolo pode aparecer, **exceto** o LEMON. Nunca há premiação se o símbolo LEMON aparecer em qualquer uma das rodas (mesmo que você tire LEMON em todas as três rodas). Qualquer outra configuração que não seja uma das configurações vencedoras, também não tem premiação nenhuma.

Escreva um programa que simula a máquina caça-níquel. Seu programa deve considerar que o usuário tem um crédito de 50 dólares e deixar que o usuário jogue até que o dinheiro acabe

ou que o usuário decida parar. Durante cada rodada seu programa deve cobrar 1 dólar, simular a rotação das rodas, avaliar o resultado e pagar o usuário se ele conseguir alguma configuração vencedora. Por exemplo, um usuário poderia ser sortudo o suficiente para ter o seguinte jogo:

```
Você gostaria de instruções (S/N)? N
Você tem $50. Você gostaria de jogar (S/N)? S
PLUM      LEMON      LEMON      -- Você perdeu
Você tem $49. Você gostaria de jogar (S/N)? S
BAR       PLUM       PLUM       -- Você perdeu
Você tem $48. Você gostaria de jogar (S/N)? S
CHERRY   CHERRY   ORANGE    -- Você ganhou $5
Você tem $52. Você gostaria de jogar (S/N)? S
LEMON    ORANGE    BAR       -- Você perdeu
Você tem $51. Você gostaria de jogar (S/N)? S
BELL     BELL     BELL     -- Você ganhou $20
Você tem $70. Você gostaria de jogar (S/N)? S
ORANGE   PLUM     BELL     -- Você perdeu
Você tem $69. Você gostaria de jogar (S/N)? S
BAR      BAR      BAR      -- Você ganhou $250
Você tem $318. Você gostaria de jogar (S/N)? N
```

Nesta simulação você deve fazer com que cada um dos seis símbolos em uma roda tenham a mesma probabilidade de ser sorteado (na vida real isso não é assim pois o caça-níquel não traria lucro ao cassino: na verdade a probabilidade de cada símbolo é cuidadosamente ajustada na máquina para fazer com que o apostador ganhe algumas vezes uma pequena quantidade de dinheiro e perca muito mais para o cassino lucrar).

Se o usuário solicitar instruções no início do jogo, seu programa deve exibir um texto informando como o usuário deve jogar.

- (d) Escreva duas implementações diferentes para a função “`strcmp`” da interface `string.h` da Biblioteca C Padrão, que trabalhem diretamente com a representação interna das strings. Suas implementações não podem chamar nenhuma função que trabalhe com strings. Uma deve considerar a string como um array de caracteres e a outra deve considerar as strings como um ponteiro para um caractere.
- (e) Crie uma função chamada de “`capitalizar`”, que recebe uma string qualquer e retorne uma string cuja primeira letra esteja em maiúscula (se for uma letra) e todas as outras letras estejam em minúsculas. Caracteres que não sejam letras não devem ser afetados. Por exemplo: `capitalizar("BOOLEANO")` e `capitalizar("booleano")` deve retornar “Booleano”. **Atenção:** devido à dificuldade de trabalhar com caracteres acentuados em C sem o uso de tipos de dados e funções especiais, seu programa pode considerar apenas os caracteres da tabela ASCII.
- (f) Um **palíndromo** é uma palavra que é lida da mesma maneira de trás para frente, por exemplo: ovo, ana, radar, renner, ata, esse, asa, ama, oco, rir, reter, etc. Escreva um programa que tem um predicado “`e_palindromo(str)`”, que retorna TRUE se a string `str` for um palíndromo, e FALSE caso contrário. Além disso, projete e escreva um programa de teste para demonstrar que seu predicado funciona. **Atenção:** devido à dificuldade de trabalhar com caracteres acentuados em C sem o uso de tipos de dados e funções especiais, seu programa pode considerar apenas os caracteres da tabela ASCII. Assim a palavra “aná” será considerada um palíndromo pois o usuário deve informar sem o acento na segunda letra “a”.

- (g) Uma das formas mais simples de criptografia são as **cifras de substituição de letras**, nas quais cada uma das letras de uma mensagem inicial (chamada de **texto puro**) é substituída por alguma letra diferente na versão criptografada da mensagem (chamada de **texto cifrado**). Um tipo particularmente simples de cifra de substituição de letras é a **cifra cíclica**, na qual cada letra é substituída por outra letra que está a uma distância fixa considerando a ordem alfabética. A palavra “cíclica” refere-se ao fato de que se na operação de substituição a letra cifrada for ultrapassar a letra Z, você simplesmente faz a volta e começa de novo a partir da letra A. A distância fixa que será utilizada para substituir as letras é chamada de **chave**. Por exemplo: o texto puro “ZEBRA” seria cifrado para o texto “DIFVE” se usarmos uma chave de tamanho 4.

Crie a função “`cifrar_string(str, chave)`”, que recebe uma string representando o texto puro e um inteiro representando a chave e retorna uma nova string, o texto cifrada, onde cada letra na string original é substituída pela letra que está a uma distância dada pela chave. Por exemplo: se a chave é 6 e a letra original é “A”, a letra cifrada será “G”; se a chave é 6 e a letra original é “W”, a letra cifrada será “C”. Note que a **chave pode ser negativa** e, nesse caso, a distância é contada de trás para frente.

Escreva uma programa de teste semelhante ao seguinte:

```
Este programa criptografa mensagens utilizando uma cifra cíclica.
Para parar, informe 0 como a chave.

Informe a chave: 4
Entre o texto puro: ABCDEFGHIJKLMNOPQRSTUVWXYZ
O texto cifrado é : EFGHIJKLMNOPQRSTUVWXYZABCD

Informe a chave: 13
Entre o texto puro: Esta e uma mensagem secreta.
O texto cifrado é : Rfgn r hzn zrafntrz frpergn.

Informe a chave: -1
Entre o texto puro: IBM-9000.
O texto cifrado é : HAL-9000.

Informe a chave: 0
```

Note que a operação de cifragem aplica-se somente às **letras não acentuadas**. Quaisquer outros caracteres como pontuação, números ou letras acentuadas não devem ser modificados. Note também que letras maiúsculas no texto puro devem ser substituídas por letras cifradas também maiúsculas.

- (h) Usando apenas funções da interface `string.h` (não use a biblioteca `CRpaic` ou a `CSLIB`), implemente a função “`sub_string(str, p1, p2)`”, que retorna uma sub-string de `str` começando na posição `p1` e terminando na posição `p2`. Sua função deve funcionar de acordo com as seguintes regras:
- Se `p1` for negativo, deve ser considerado como 0, indicando o primeiro caractere da string;
 - Se `p2` for maior do que “`strlen(str) - 1`”, considere que `p2` deve ter o valor de `strlen(str)`, indicando o último caractere da string; e
 - Se `p1` for maior do que `p2`, a função deve retornar a string vazia.
- (i) Escreva o programa “`wc .c`”, que lê um arquivo e informa quantas linhas, palavras e caracteres estão nesse arquivo. Para os propósitos deste programa, uma palavra consiste de uma seqüência consecutiva de quaisquer caracteres, exceto caracteres que representem espaços em branco.

Por exemplo: se o arquivo `apologia_sem_acentos.txt` tiver o seguinte trecho do livro “Apologia de Sócrates”, de Platão, sem acentos

Mas tambem vos, o juizes, deveis ter boa esperanca em relacao a morte, e considerar esta unica verdade: que nao e possivel haver algum mal para um homem de bem, nem durante a vida, nem depois da morte, e que os Deuses nao se interessam do que a ele concerne; e que, por isso mesmo, o que hoje aconteceu, no que a mim concerne, nao e devido ao acaso, mas e a prova de que para mim era melhor morrer agora e ser libertado das coisas deste mundo. Eis tambem a razao porque a divina voz nao me dissuadiu, e porque, de minha parte, nao estou zangado com aqueles cujos votos me condenaram, nem contra meus acusadores.

Nao foi com esse pensamento, entretanto, que eles votaram contra mim, que me acusaram, pois acreditavam causar-me um mal. Por isto e justo que sejam censurados. Mas tudo o que lhes peço e o seguinte: Quando os meus filhinhos ficarem adultos, atormentai-os como eu os vos atormentei, quando vos parecer que eles cuidam mais de riquezas e de honrarias do que da Verdade. E, se acreditarem ser qualquer coisa nao sendo nada, reprovai-os, como eu a vos: nao vos preocupeis com aquilo que nao lhes e devido.

E, se fizerdes isso, terei de vos o que e justo, eu e os meus filhos.

E a hora de irmos: eu para a morte, vos para as vossas vidas; quem tera a melhor sorte? So os Deuses sabem.

seu programa deve ser capaz de reproduzir o seguinte comportamento:

```
Arquivo: apologia_sem_acentos.txt
Linhas:      27
Palavras:    243
Caracteres:  1296
```

(j) Projete e implemente a interface “`cartas.h`”, que exporta as seguintes coisas:

- Um tipo “`valor_t`”, que permite que você represente o valor de uma carta. Os valores desse tipo devem incluir os inteiros entre 2 e 10, mas também devem incluir as constantes AS (A), VALETE (J), DAMA (Q) e REI (K).
- Um tipo “`naipe_t`”, consistindo dos quatro naipes do baralho: COPAS (C), OUROS (O), ESPADAS (E), PAUS (P).
- Um tipo “`carta_t`” que combina um valor e um naipe.
- Uma função “`nova_carta(valor, naipe)`”, que cria uma `carta_t` a partir do valor e naipe.
- Duas funções, `valor(cart)` e `naipe(cart)`, que permitem que o cliente consulte o valor e o naipe de uma carta. Obs.: essas duas funções poderiam facilmente ser substituídas diretamente por um código que selecionasse essas informações diretamente dos componentes apropriados de uma carta, mas criar funções significa que o cliente não

precisa prestar atenção à estrutura interna subjacente do tipo de dados.

- Uma função “nome_carta(cartas)” que retorna uma string representando a carta. Essa string deve começar com o valor da carta, mostrado como uma letra no conjunto “A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, R”, seguido por uma letra no conjunto “C, O, E, P”. Essa string geralmente terá duas letras (ex.: 3P), mas poderá ter três letras (ex.: 10C).

- (k) Usando a interface `cartas.h` do exercício anterior, crie um programa que inicialize um baralho completo de 52 cartas, embaralhe as cartas e depois mostre as cartas embaralhadas, como o seguinte exemplo:

AC	10P	5O	4C	JE	AO	KC	3P	4P	2O	6P	AP	JO
2C	KE	9C	5E	AE	6E	6O	8E	KO	2E	7C	8C	5P
8P	QC	4E	9E	QE	9O	6C	7E	9P	7O	3C	JC	10O
KP	10C	8O	2P	7P	QO	JP	5C	QP	4O	10E	3O	3E

Uma das maneiras mais fáceis de embaralhar o conteúdo de um array é adotar a estratégia representada pelo seguinte pseudo-código:

```
para (cada posição P1, no array)
{
    Escolha uma posição aleatória P2 entre P1 e o fim do array
    Troque os valores nas posições P1 e P2
}
```

- (l) Escreva um programa para jogar o Jogo da Forca. No Jogo da Forca, o computador começa selecionando uma palavra secreta aleatoriamente a partir de uma lista de palavras. Então o computador imprime uma lista de hifens — um para cada letra da palavra secreta — e solicita que o usuário escolha uma letra. Se a letra que o usuário escolheu aparecer na palavra secreta, a palavra secreta é reexibida mostrando todas as letras que o usuário escolheu na palavra secreta, nas posições corretas, juntamente com todas as letras escolhidas corretamente em rodadas anteriores. Se a letra que o usuário escolheu não aparecer na palavra, o jogador perde uma “vida”. O usuário continua a escolher letras até que: 1) ele consiga acertar todas as letras e revelar a palavra secreta; ou 2) ele perca todas as suas “vidas”. Neste programa o usuário começará com 8 vidas.

Define e implemente uma interface chamada de “`velha.h`” que exportará duas funções com os seguintes nomes: `inicializar_dicionario` e `sortear_palavra`.

A função `inicializar_dicionario` recebe como argumento um arquivo texto contendo uma lista de palavras, uma palavra por linha, e carregará essas palavras em um array declarado como variável global estática na implementação da interface.

A função `sortear_palavra` não recebe nenhum argumento e retorna uma palavra escolhida aleatoriamente a partir desse array interno de palavras.

Seu programa deve se comportar conforme o exemplo na próxima página.

Bem-vindo ao Jogo da Forca!
Eu irei escolher uma palavra secreta. Em cada rodado você irá "chutar" uma letra. Se a letra estiver na palavra secreta, eu mostrarei onde ela está; se a letra não estiver na palavra secreta, uma parte de seu corpo ficará pendurada na forca. Seu objetivo é acertar a palavra secreta antes de você seja enforcado (você tem 8 chances).

A palavra secreta é essa: -----
Você tem 8 chance(s) restante(s).
Escolha uma letra: O
Você acertou!

A palavra secreta é essa: -O-----O-
Você tem 8 chance(s) restante(s).
Escolha uma letra: F
Não há nenhuma letra F na palavra secreta.

A palavra secreta é essa: -O-----O-
Você tem 7 chance(s) restante(s).
Escolha uma letra: M
Você acertou!

A palavra secreta é essa: -OM-----O-
Você tem 7 chance(s) restante(s).
Escolha uma letra: G
Não há nenhuma letra G na palavra secreta.

A palavra secreta é essa: -OM-----O-
Você tem 6 chance(s) restante(s).
Escolha uma letra: A
Você acertou!

A palavra secreta é essa: -OM---A-O-
Você tem 6 chance(s) restante(s).
Escolha uma letra: P
Você acertou!

A palavra secreta é essa: -OMP--A-O-
Você tem 6 chance(s) restante(s).
Escolha uma letra: S
Não há nenhuma letra S na palavra secreta.

A palavra secreta é essa: -OMP--A-O-
Você tem 5 chance(s) restante(s).
Escolha uma letra: C
Você acertou!

A palavra secreta é essa: COMP--A-O-
Você tem 5 chance(s) restante(s).
Escolha uma letra: U
Você acertou!

A palavra secreta é essa: COMPU-A-O-
Você tem 5 chance(s) restante(s).
Escolha uma letra: T
Você acertou!

A palavra secreta é essa: COMPUTA-O-
Você tem 5 chance(s) restante(s).
Escolha uma letra: D
Você acertou!

A palavra secreta é essa: COMPUTADO-
Você tem 5 chance(s) restante(s).
Escolha uma letra: R
Você acertou!

Você acertou a palavra secreta: COMPUTADOR.
Você ganhou o jogo, parabéns!