*Alguns Exercícios de Programação de em Linguagem C

Nelma Moreira

Departamento de Ciência de Computadores
Faculdade de Ciências, Universidade do Porto
email: nam@ncc.up.pt

1997

1 Execução de algumas instruções do C

1. Suponha que inicialmente `x` tem o valor `3` e `y` o valor `10`. Qual o valor das variáveis `x` e `y` após a execução das seguintes instruções:

   (a) `x=x+y;`

   (b) `x=x+x;`

   (c) `x=y; x=3;`

   (d) `x=x*x; x=x+x;`

   (e) `y=x; x=y;`

   (f) `y+=x++;`

   (g) `y-=++x;`

2. O que está errado com cada uma das seguintes instruções em C?

   (a) `x=x+y`

   (b) `x+y=5;`

*Baseados no apontamentos *Introdução à Programação* de Armando B. Matos, 1988

(c) `if x>y x=5;`

(d) `if (x>y) { x=5 }`

(e) `if (x>y) x=5 else x=6;`

3. Para cada um dos programas seguintes:

   (a) Siga o funcionamento para os dados indicados, fazendo uma tabela da variação dos valores das variáveis.

   (b) Indique informalmente a sua função genérica.

   (a) `20, 30`

```
#include<stdio.h>

main() {
  int x,y,m;
  printf("Introduza dois valores: ");
  scanf("%d %d",&x,&y);
  if (y<x) m=x; else m=y;
  printf("%d \n",m);
}
```

   (b) `-2`

```
#include<stdio.h>

main() {
  int x,m;
  printf("Introduza um valor: ");
  scanf("%d",&x);
  if (x<0) m=-x; else m=x;
  printf("%d \n",m);
}
```

   (c) `3, 9, 7`

```
#include<stdio.h>

main() {
  int x,y,z,m;
  printf("Introduza tres valores: ");
  scanf("%d %d %d",&x,&y,&z);
  m=x;
```

```
  if (y<=z){
    if (x<=y) {m=y;}
    else if (z<=x) {m=z;}
  } else
    if (x<=z) {m=z;}
    else if (y<=x) {m=y;}
  printf("%d \n",m);
}
```

(d) 10

```
#include<stdio.h>

main() {
  int s=0,i=1,n;
  printf("Introduza um valor: ");
  scanf("%d",&n);
  while (i<=n) {
    s=s+i*i;
    i=i+1;
  }
  printf("%d \n",s);
}
```

(e) 33, 4, 8, 4, 5, 4, 7, 33

```
#include<stdio.h>

main() {
  int x,y,n,s=0,;
  scanf("%d",&n);
  scanf("%d",&x);
  scanf("%d",&y);
  s=0;
  while (y!=n) {
    if (y=x) s=s+1;
    scanf("%d",&y);
  }
  printf("%d \n",s);
}
```

(f) 20

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

```
main() {
 int n;
 while(n<15) n=n-1;
 while(n>10) n=n+1;
 printf("%d \n",n);
}
```

(g) 325

```
#include<stdio.h>

main() {
 int n,x;
 scanf("%d",&n);
 while(n!=0) {
  x=n/10;
  printf("%d \n",n-10*x);
  n=x;
 }
}
```

(h) 325

```
#include<stdio.h>
#define Base 10

main() {
   int n,s;
   scanf("%d",&n);
   while(n!=0) {
     s=s+n%Base;
     n=n/Base;
   }
   printf("%d \n",s);
}
```

(i) 15, 2

```
#include<stdio.h>

main() {
 int n,b;
 scanf("%d %d",&n, &b);
```

```
 while(n>=b) {
  printf("%d \n",n%b);
  n=n/b;
 }
  printf("%d \n",n);
}
```

(j)
```
#include<stdio.h>

#define MAX 20
main() {
  int x=0,y=1,z,n;
  printf("%d \n", x);   printf("%d \n", y);
  while (y<=MAX) {
    z=x+y;
    x=y;
    y=z;
  printf("%d \n",y);
  }}
```

4. Os seguintes programas pretendiam resolver os problemas enunciados mas nem todos funcionam correctamente. Corriga os erros e teste a sua execução.

   (a) Somar os números ímpares de `1` a `999`.

```
#include<stdio.h>

main() {
  int s, k=1;

  while (k!=1000) {
    s=0;
    s=s+k;
    k=k+2;
  }
  printf("a soma e %d \n",s);
}
```

   (b) Determinar o menor múltiplo comum entre `a` e `b` inteiros positivos.

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

```
main() {
  int a,b,m;

  m=a; y=0;
  while (y==0) {
    if (m%b==0) y=1; else m=m+a;
  }
  printf("o m.m.c entre %d e %b e %d \n",a,b,m);
}
```

(c) Determinar por subtracções sucessivas o quociente inteiro entre `a` e `b`, com `b` $\neq 0$.

```
#include<stdio.h>

main() {
  int a,b,q;
  scanf("%d %d",a,b);
  q=0;
  while (a>b) {
   q++;
   a-=b;
 }
 printf(¨o quociente e %d\n¨,q);
}
```

(d) Determinar a soma das primeiras `n` potências de `k`.

```
#include<stdio.h>

main() {
  int k, n, s=1;

  scanf(¨%d¨, k);
  while(n!=0) {
   s=k;
   k=k*k;
   n=n-1;
  }
  printf(¨a soma e %d¨,s);
}
```

2 Problemas simples com inteiros

Escreva programas em linguagem C para cada um dos seguintes problemas.

1. Dados três inteiros positivos
   (a) determinar se podem ser os comprimentos dos lados de um triângulo rectângulo;
   (b) determinar o menor;
   (c) a soma dos seus quadrados;
   (d) o quadrado da sua soma;
   (e) a sua média aritmética.
2. Repetir o exercício anterior, com a excepção da primeira alínea, para `n` inteiros positivos.
3. Dada uma sequência de inteiros terminada por zero, determinar o número de números pares.
4.
   (a) Determinar os múltiplos de 5, não múltiplos de 3 entre 100 e 10000.
   (b) Determinar o menor inteiro positivo que dividido por 6 dá resto 5 e dividido por 11 dá resto 8.
5. Dados dois inteiros `m` e `n` determinar se, na base `b`, os algarismos de `m` coincidem com os algarismos de `n` por ordem inversa.
6. Calcular o preço de saldo de um artigo, sabendo que os descontos em função do preço actual `P` são os seguintes:

Preço		Desconto
$\texttt{P} > 10000$		40%
$5000 < \texttt{P} \leq 10000$		20%
$2500 < \texttt{P} \leq 5000$		10%
`outros`		5%

7.
   (a) Determinar todos os múltiplos de 5 da forma $2^n + 1$, inferiores a 1000.
   (b) Determinar todos os primos a forma $2^n + 1$, inferiores a 10000.

8. Um número inteiro não negativo diz-se *perfeito* se é igual à soma dos seus divisores próprios. Por exemplo, 6=2+3+1. Dado `n`, determine todos os números perfeitos inferiores a `n`.

9. Aproveitando o algoritmo desenvolvido na alíena anterior, calcular a percentagem dos números entre 2 e 20000 tal que:

   (a) são inferiores à soma dos seus divisores;

   (b) são iguais à soma dos seus divisores;

   (c) são superiores à soma dos seus divisores.

10. Dada uma sequência de $\texttt{n} \geq \texttt{1}$ inteiros determinar o número de subsequências não decrescentes.

11. Determinar numa sequência de valores lidos o número de valores maiores do que os seus dois vizinhos. Por exemplo, para

    ```
    8, 2, 4, 1, 6, 12, 5, 9
    ```

    `4` e 12 verificam essa condição.

12. Produzir uma tabela com 3 colunas, a primeira contendo os inteiros de 1 a 10, a segunda os respectivos quadrados e a terceira os respectivos cubos.

13. Imprimir os termos inferiores a 10000 da sucessão assim determinada: os 3 primeiros termos são dados; cada termo seguinte é a soma dos 3 termos anteriores.

14. Dado um ano e uma data em dias, calcular o dia do mês e o mês correspondente. Dado um ano (> 1900) e um mês imprimir o calendário desse mês. Suponha que o 1 de Janeiro de 1900 é uma segunda-feira.

15. Converter um tempo em segundos para horas, minutos e segundos. E vice-versa.

16.

3 Mais alguns problemas numéricos

1. Imprimir os valores das funções `seno(x)`, `coseno(x)` e `tangente(x)` para `x` em graus e $\texttt{0} \leq \texttt{x} \leq \texttt{90}$ com intervalos de 10.

2. Dado `x`, determinar $\mid x - 5 \mid$.

3. Dado `n` > 0 inteiro, calcular

$$\sum_{i=1}^{n} 1/2^i$$

4. Dado x e n calcule:

   (a) $1 + x + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!}$

   (b) $x - \frac{x^3}{3!} + \ldots + \frac{(-1)^{2n+1}x^{2n+1}}{(2n+1)!}$

5. Dado `n`, calcular `k`

$$k = \frac{1}{1+2} + \frac{1}{2+3} + \frac{1}{3+4} + \ldots + \frac{1}{n+(n+1)}$$

6. A sucessão

$$a_n = 4(1 - 1/3 + 1/5 - \ldots + (-1)^{n+1}1/(2n-1))$$

   converge para π. calcule a_{1000} fazendo sair os resultados para valores de `n` de 100 em 100.

7. Tabelar, com 6 algarismos, a função

$$\texttt{f(x)} = \cos(\sqrt{\texttt{x}} + \texttt{1})$$

   para `x` $=$ `0(5)45`, em graus.

8. Tabelar a função

$$\texttt{f(x)} = (-\texttt{1})^{(\texttt{1/x})}/(\texttt{1/x})!$$

   para `x` $=$ `0.1(0.05)1.0`.

9. Dados os números reais `a`, `b` e `c` determine as raízes da equação de segundo grau $\texttt{ax}^2 + \texttt{bx} + \texttt{c} = 0$ indicando se as raízes são reais ou complexas.

4 Variáveis Indexadas (*Arrays*)

Escreva programas em linguagem C para cada um dos seguintes problemas, utilizando se necessário variáveis indexadas.

1. Dada uma variável indexada de inteiros `a[100]` construa e imprima uma variável indexada `b[100]` tal que:
   - os elementos de ordem par são iguais aos de `a` divididos por 2.
   - os elementos de ordem ímpar são iguais aos de `a` multiplicados por 2.
2. Lidos o valor de `n` e valores `a[0]`, `a[2]`, ..., `a[n-1]`, construa e imprima a variável indexada resultante de cada uma das seguintes operações (usando apenas a variável indexada `a[]`):
   (a) Efectuar uma permutação circular tal que `a[n-1]` passe para `a[0]`;
   (b) Deslocar para a esquerda os últimos `n-1` valores;
   (c) Deslocar os primeiros `n-1` valores uma posição para a direita;
   (d) Inverter a posição dos valores de `a[]`;
   (e) Substituir cada `a[i]` pela média `(a[i-1]+a[i]+a[i+1])/3`.
3. Dada uma sequência de `n` números, eventualmente repetidos, determinar o máximo da diferença de 2 elementos consecutivos assim como a(s) posição(ões) do primeiro desses elementos na sequência.
4. Dada uma variável indexada `v[]` com valores inteiros, pretende-se imprimir a soma máxima de uma subsequência de elementos contíguos. Indicar também os índices inicial e final da subsequência.
5. Use a função `rand()` para gerar 1000 úmeros aleatórios entre 1 e 100. Conte o número de ocorrências de cada número e imprima-os por ordem decrescente de frequência.
6. Dado um número inteiro `n` e uma base de numeração `b` construa uma variável indexada que contenha a representação de `n` na base `b`.
7. Duas variáveis indexadas de inteiros `n[]` e `m[]` contêm os dígitos da representação de dois números respectivamente `n` e `m` numa base `b`. Calcular a soma desses dois números nessa base e converter o resultado para a base `10` e imprimir.
8. Análogo ao anterior, mas supondo que a base é 2. Calcular a soma e o produto desses dois números na base 2.

9. Dadas duas sequências de números (sem repetições) `a` e `b`, ordenadas por ordem crescente construir uma nova sequência `c`, também ordenada, por junção dos elementos das duas dadas, eliminando repetições.

10. Ordenar por ordem crescente uma sequência de valores inteiros `a[n]` usando o método da:

    Inserção Introduzir cada elemento `a[i]`, na subsequência ordenada de `a[0]`, ..., `a[i-1]` de modo a manté-la ordenada.

    ```
    Para i=1,... n-1
      faça x=a[i]
      inserir x na posição correcta entre as posições 0 e i-1
    ```

    Para inserir `x` na posição correcta pode usar os métodos de pesquisa

    - linear
    - binária

    Bolha Em cada iteração, percorrer os elementos da sequência e comparar pares de elementos consecutivos trocando-os se não estiverem ordenados. Ao fim de uma iteração o elemento maior é encontrado. Repetir o processo até todos os elementos estarem ordenados (no máximo `n` vezes).

    ```
    Para i=0,... n-2 e enquanto houver trocas
      para j=1,... n-i
        se a[j]>a[j+1} trocar a[j] com a[j+1]
    ```

11. Pretende-se determinar e imprimir numa sequência de `n` valores lidos, todos eles compreendidos entre `0` e `150`, o número de valores entre `0` e `5`, o número de valores entre `5` e `10`, ..., o número de valores entre `145` e `150` (o limite inferior é excluído e o superior é incluído).

12. Lidos o valores `n`, e `x[i]` e `y[i]` para `i=0,1,...n-1`, pretende-se determinar o par de pontos mais próximo, admitindo-se que cada par `(x[i],y[i])` representa um ponto no plano.

13. O triângulo de Pascal é caracterizado por ter em cada linha:

    (a) na primeira linha o número `1`;

    (b) nas outras a soma entre o número imediatamente acima e o número acima e à esquerda.

    Começa assim

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
...
```

Desenvolva um algoritmo (usando apenas uma variável uni-dimensional) que imprima as primeiras `20` linhas do triângulo de Pascal.

Nota: numerando as linhas e colunas de `0` a `n-1`, na linha `i` e coluna `j`, encontra-se o número de combinações de `i` objectos tomados `j` a `j`.

4.1 Cadeias de Caracteres (*Strings*)

1. Sem utilizar as funções das bibliotecas *standard* do `C` escreva funções em `C` para:
   (a) determinar o comprimento duma *string*;
   (b) copiar uma string para outra;
   (c) verificar se duas *strings* são iguais;
   (d) concatenar duas *strings*;
   (e) dadas duas *strings* determinar se a primeira é uma subsequência da segunda;
2. Ler uma linha do terminal e remover os caracteres `brancos` (espaço e tabulação) e de pontuação.
3. Determinar a maior linha dum texto (*string*) lido, guardar a maior linha e escrevê-la.
4. Uma *string* é uma *palavra* se não contiver caracteres `brancos`. Lido um texto:
   (a) Determinar a maior palavra e imprimi-la.
   (b) Classificar as palavras dum texto quanto ao comprimento em 5 classes e determinar a frequência absoluta de cada classe (número de palavras em cada classe). Considere na primeira classe comprimentos de palavras entre (exclusive) `0` e `3`, na segunda entre `3` e `6` , terceira entre `6` e `9`, na quarta entre `9` e `12` e na quinta, comprimentos maiores que `12`.
   (c) Modifique o programa anterior para imprimir os resultados em forma de histograma:

```
Comprimento de Palavras        Frequência
          0--3                   ******
          3--6                  **********
           ...          ...
```

(d) Suponha dada uma tabela de palavras, denominadas *palavras-chave*. Determine o número de ocorrências de cada *palavra-chave* no texto, fazendo sair os resultados da seguinte forma:

```
Palavra-Chave     Ocorrências
     ...              ...
```

5. Formatação de um texto. Ler um texto, guardá-lo e formatá-lo segundo as seguintes opções:

   (a) número máximo de caracteres por linha: p.e. 80.

   (b) ajustamento do texto: à esquerda, à direita ou ajustado igualmente à esquerda e à direita.

   No início o utilizador deve poder escolher as características anteriores. O texto formatado (que deve ser guardado noutra variável indexada) deve ser impresso no final. Para ajustar o texto, pode introduzir ou retirar caracteres brancos.

6. Baseado no método de ordenação por inserção, e usando as funções da biblioteca do `C` `gets()`, `puts()`, `strcpy()` e `strncmp()`, ordene uma sequência de linhas lidas do terminal. Considere uma variável bidimensional `texto[L][C]` para guardar as linhas de modo ordenado. Sempre que uma linha é lida ela é inserida em `texto`, de modo a que a variável fique ordenada por linhas.

4.2 Variáveis Indexadas Multidimensionais

1. Dada uma matriz `a[n][m]` determinar a matriz que resulta de:

   (a) Trocar o menor elemento de cada linha `i` pelo elemento maior;

   (b) Deslocar a primeira coluna para a segunda, a segunda para a terceira, ..., a `m`-ésima para a primeira;

   (c) ordenar a matriz considerando o seguinte critério: a linha `i` é maior do que a linha `j` se na coluna de menor índice em que os elementos diferem, o elemento da linha `i` é maior do que o da linha `j`.

2. Dadas duas matrizes de inteiros `a[n][m]` e `b[m][k]` calcular a matriz `c[n][k]` correspondente ao produto matricial de `a` por `b`.

3. Dada uma matriz quadrada de inteiros `a[n][n]`

(a) calcular a sua transposta.

(b) substituir cada componente de `a[][]` não pertencente aos limites (4 cantos: superior, inferior, esquerdo e direito) pela média aritmética dos seus `8` vizinhos.

4. Dada uma matriz com 1000 linhas e 7 colunas representando 1000 sorteios do totoloto (para valores inteiros de 1 a 52) calcular:

   (a) Para cada sorteio o número de pares de números consecutivos;

   (b) Os 3 números que ocorrem mais vezes;

   (c) A maior sequência de números consecutivos.

5. Simulação do jogo do galo. O jogo do galo joga-se num tabuleiro de 3x3 e dois jogadores. Cada jogador coloca alternadamente um `1` ou um `2` numa das quadriculas e ganha o que colocar 3 peças em linha (horizontal, vertical ou diagonal principal). O tabuleiro é representado por uma variável indexada `a[2][2]` que contém as jogadas feitas, supondo-se que se o jogador `i` jogou numa certa posição então o conteúdo dessa posição é `i`, para `i=1,2`. Se uma posição ainda não foi jogada o conteúdo de `a` nessa posição é `0`. Nenhum jogador pode jogar numa posição já preenchida e o jogo termina empatado se todas as posições estão ocupadas e nenhum jogador ganhou. Em cada jogada deve ser imprimido o tabuleiro, isto é, `a`. No início o programa deve pedir o nome de cada um dos jogadores e imprimir o nome do jogador que ganhar.

6. Simulação do jogo *5 em linha*.

   Objectivo do Jogo Dois jogadores preenchem alternadamente as posições dum tabuleiro $N \times N$, $N > 4$. Um joga com *peças* `0` e o outro joga com *peças* `1`. Ganha o jogo, o primeiro jogador que conseguir colocar 5 peças consecutivas na mesma direcção: numa linha, coluna ou diagonal (faz *5 em linha*). O jogo termina – empatado – se já não houverem posições para preencher com peças. Supõe-se que inicialmente todas as posições do tabuleiro contêm o símbolo `X`.

   Regras do Jogo A situação inicial do tabuleiro e a numeração das linhas e colunas é:

	1	2	3	4	5
1	X	X	X	X	X
2	X	X	X	X	X
3	X	X	X	X	X
4	X	X	X	X	X
5	X	X	X	X	X

Os dois jogadores jogam alternadamente. Em cada jogada, um jogador selecciona uma posição da matriz indicando apenas a coluna c (de 1 a N) em que pretende jogar. Se a coluna c estiver toda preenchida, e, ainda restarem posições livres noutra coluna, terá de escolher uma dessas colunas *livres*. A posição em que ficará a sua peça será a corresponde à linha de maior numeração ainda *livre*, nessa coluna. Isto é, para uma dada coluna c, a primeira posição a ser preenchida é a (c, N), a segunda a $(c, N-1)$, a terceira a $(c, N-2)$, ... No fim de cada jogada, o computador terá de avaliar se o jogador ganhou o jogo. Para isso, basta verificar se com a peça que jogou conseguiu fazer *5 em linha*. Note que não é necessário percorrer todas as posições do tabuleiro, mas apenas as posições que estão na mesma linha, coluna ou diagonais da última peça jogada.

O programa a desenvolver deve, em cada jogada:

- mostrar o tabuleiro
- indicar qual o jogador que deve jogar
- pedir ao jogador que seleccione uma coluna
- verificar se o jogador fez *5 em linha*
- verificar se o jogo terminou e indicar o vencedor
- permitir a continuação do jogo

5 Estruturas, Apontadores e Listas Ligadas

1. (**Manipulação duma base de dados de livros**) Suponha que para cada livro existe informação sobre os seguintes campos:

   `Título` 80 caracteres

   `Autor1` 20 caracteres (Nome Próprio)

   `Autor2` 20 caracteres (Apelido)

   `Ano de Edição` inteiro sem sinal

   `Tema` 40 caracteres

   (a) Defina uma estrutura em `C` para guardar a informação anterior e defina uma variável indexada que contenha 100 apontadores para essas estruturas.

   (b) Escreva funções que permitam:

   i. Introduzir um novo livro (pelo terminal);

   ii. Leitura dos dados de um ficheiro, supondo que no ficheiro cada campo é guardado numa linha; para optimizar as pesquisas pode guardar a informação ordenada (lexicograficamente) pelo campo `Autor2`;

   iii. Retirar a informação de um livro da base de dados;

   iv. Guardar a informação da base de dados num ficheiro;

   v. Procurar um livro por: `Título` ou `Autor2`;

   vi. Produzir os seguintes relatórios:

   A. listagem de todos os livros

   B. listagem de todos os livros de um autor;

   C. listagem de todos os livros de um tema;

   D. listagem de todos os livros editados num mesmo ano;

   Para cada um dos relatórios, deve ser pedido ao utilizador para seleccionar quais os campos que pretende que sejam listados.

   (c) Escreva um interface que após lida a base de dados, usando a função definida em 1(b)ii, permita ao utilizador selecionar uma das tarefas:

```
1. Introduzir novo livro
2. Procurar por autor
3. Procurar por titulo
4. Retirar um livro;
5. Relatórios
6. Terminar
```

Se for selecionada a tarefa `5.` o utilizador ainda deverá escolher qual dos relatórios e qual a informação a imprimir. Se for selecionada a tarefa `6.`, deverá ser chamada a função que guarda a base de dados em ficheiro.

2. Modifique o programa anterior, supondo que a base dados é guardada numa lista ligada. Sugestão: Use a seguinte estrutura para cada livro:

```
struct livros {
    char titulo[80];
    char autor1[20];
    char autor2[20];
    unsigned int ano;
    char tema[40];
    struct livros *prox;
};
```

3. Considerando uma `fila` escreva uma função para determinar o seu número de elementos. Suponha

   (a) a `fila` implementada usando memória sequêncial, como uma lista circular;

   (b) a `fila` implementada usando memória dinâmica

4. Dadas duas `filas`, implementadas com listas ligadas, escreva um programa que junte as duas filas numa, alternando os elementos das duas primeiras. Se uma das filas for maior, os seus elementos serão colodados no fim.

5. Uma matriz bi-dimensional de inteiros *esparsa* é uma matriz de grandes dimensões em que a maioria das suas entradas são nulas. Uma tal matriz pode ser representada convenientemente usando uma lista ligada onde cada elemento tem 5 campos: a linha; a coluna; o valor; a linha do proximo elemento não nulo; a coluna do próximo elemento não nulo.

   (a) Escreva uma função que dada uma matriz esparsa construa uma lista ligada equivalente;

   (b) Escreva uma função que construa uma variável unidimensional em que cada elemento é a soma de todos os elementos duma mesma linha da matriz esparsa;

   (c) Escreva uma função que adicione duas matrizes esparsas (das mesmas dimensões) usando listas ligadas;

6. Crie, usando a função `rand()`, uma árvore binária com 30 nós em que cada nó contém um inteiro entre 1 e 100. Para cada nó, a sub-árvore esquerda contém nós cujos inteiros são menores que o desse nó e a sub-árvore direita contém nós cujos inteiros são maiores.

   (a) escreva uma função que determine o valor maior na árvore;

   (b) escreva uma função que determine o valor menor na árvore;

   (c) escreva uma função que dados dois inteiros, imprima todos os inteiros da árvore entre esses valores.

7. Utilizando uma árvore binária (de pesquisa) determine a frequência de cada palavra de um texto. Cada nó da árvore guarda para além de uma palavra, a frequência de ocorrência dessa palavra no texto.

8. Considere o problema da alínea anterior, para o caso de o texto ser um progrma escrito em linguagem `C` e considerando apenas como palavras as `palavras chave` da linguagem `C`, isto é:

```
auto      double  int       struct
break     else    long      switch
case      enum    register  typedef
char      extern  return    union
const     float   short     unsigned
continue  for     signed    void
default   goto    sizeof    volatile
do        if      static    while
```

   Suponha que estas palavras se encontram guardadas, por ordem alfabética, numa variável indexada externa `char *chaves`.