Introdução à Programação

1.Ano LCC-MIERSI
DCC - FCUP

Nelma Moreira

Aula 11

Sequências de caracteres: Strings

* As **strings** são variáveis indexadas de caracteres, com características especiais:
  * ter como último elemento, o caracter de código ASCII 0, '\0', (ou pela constante simbólica NULL, definida em stdio.h).
* Este elemento marca ofim_da_sequência e geralmente **não conta** como elemento da sequência, embora conte como elemento da variável indexada.
* A biblioteca do C que manipula strings têm o cabeçalho string.h

* Declaração:

```
char <nome_de_variável>[TAMANHO]
```

* Exemplo

```
char nome[4];
nome[0]='A'; nome[1]='n';
nome[2]='a'; nome[3]='\0'
printf(“%s”,nome);
```

Strings constantes

* São sequências de caracteres entre aspas: `"ola"`
  (como as que se usam no `printf`)
* Podem usar-se para inicializar variáveis indexadas de caracteres:

```
char nome[]="Ana";
```

* Contudo a instrução seguinte não é válida

```
nome="Ana";
```

* A diferença entre `'x'` e `"x"` é que a segunda é a `string x\0`

Escrita e leitura de strings

* Nas funções `printf` e `scanf` usa-se o formato `%s`, para escrever ou ler strings:
* Se `char nome[]="Ana"`

```
printf("O nome é %s",nome);
```

* Ler uma sequência de caracteres e guardar na variável indexada `s`, colocando `0` na última posição:

```
scanf("%s",s);
```

* Nota que não se coloca o `&` antes do nome da variável!!!

* Existem outras funções de entrada e saída na biblioteca standard do C, que manipula strings:

```
gets(char s[])
```

* lê uma linha do `stdin` para a variável `s`, substituindo `\n` por `\0`;

```
int puts(char s[])
```

* escreve a string `s` e um `\n`, no `stdout`.

Funções para manipular strings

* Determinar o comprimento

```
int comp(char s[])
{ int i=0;
  while(s[i]!='\0') i++;
  return i;
}
```

* Copiar (usa-se em vez da atribuição)

```
void copia (char s[],char t[])
{
  int i=0;
  while((s[i]=t[i])!='\0')
   i++;
}
```

* Comparar (ord. lexicográfica)

```
int comparar (char s[], char
t[])
{ int i;
  for(i=0;s[i]==t[i];i++)
if(s[i]=='\0') return 0;
  return s[i]-t[i];
}
```

* Concatenar duas strings

```
void concatena (char s[], char
t[])
{ int i=0,j=0;
  while (s[i]!='\0')i++;
while((s[i++]=t[j++])!='\0');
}
```

```
int main()
{ char s[20]="ola ";
  char t[]="mundo";

  concatena(s,t);
  printf("%s tem comprimento
          %d\n",s,comp(s));
  copia(s,t);
  printf("%s\n%s\n%d\n",s,t,
       compara(s,t));
  return 0
}
```

* Executar

```
$ gcc fs.c -ofs
$./fs
ola mundo tem comprimento 9
mundo
mundo
0
```

Funções da biblioteca string

Função	Descrição
void strcpy(s1,s2)	copia s2 para s1
void strcat(s1,s2)	concatena s2 no fim de s1
int strlen(s1)	retorna o comprimento de s1
void strcmp(s1,s2)	0 se s1 é igual a s2, <0 se s1 é menor que s2 > 0 se s1 é maior que s2.

Procurar subsequência (versão do `grep`)

* Procurar uma subsequência de caracteres numa sequência de linhas.
* Sempre que a subsequência for encontrada, escrever a linha em que ela ocorreu e contar o número de ocorrências.
* Supor que a subsequência é a primeira linha lida.

* Algoritmo:
* Decidir o tamanho máximo de cada linha e da subsequência e definir variáveis indexadas de caracteres desses tamanhos:

```
char linha[MAXLINHA], seq[MAXSEQ].
```

* Ler a primeira linha e guardar na variável indexada (como string) `seq`.
* Enquanto houver linhas:
  * ler uma linha de texto, caracter a caracter e guardar na variável `linha`. Para terminar a linha lida deve ser vazia.
  * procurar a `seq` em `linha`; se encontrar escrever o número da linha e a linha.

* Como procurar a `seq` em `linha`?

1. `i=0`
2. `ver se linha[i]==seq[0]`
3. `se sim, ver se linha[i+1]==seq[1], linha[i+2]==seq[2] ,..., até seq[k]=='\0'.`
4. `Se verificar, retorna i`
5. `(Senão) Incrementar i e voltar a 2, se linha[i] != 0.`

```c
#include  <stdio.h>
#define MAXLINHA 100
#define MAXSEQ   100
int le_linha(char s[],int lim);
int procura_seq(char s[],char t[]);

int main() {
  int i=0;
  char linha[MAXLINHA],seq[MAXSEQ] = "e";
  scanf("%s",seq);
  while(le_linha(linha,MAXLINHA) > 0)
      if(procura_seq(linha,seq) >= 0) printf("%d: %s",++i,linha);
}

int le_linha(char s[],int lim) {/* lê uma linha e retorna o seu
                                   comprimento. Coloca \n no fim */
  int c, i;
  for(i = 0; i < lim-1 && (c = getchar()) != EOF && c != '\n'; i++)
    s[i] = c;
  if( c == '\n') s[i++] = c;
  s[i] = '\0';
  return(i);}

int procura_seq(char s[],char t[]){
  int i,j,k;
  for(i = 0; s[i] != '\0'; i++) {
    for(j = i, k=0; t[k]!='\0' && s[j]==t[k]; j++,k++);
    if (k > 0 && t[k] == '\0') return i;

   }
  return -1;}
```

Execução:

```
$gcc encontra.c -oenc
$enc
aaaa
skkklkasdaala;aaaa
1: skkklkasdaala;aaaa
kakskddkjk
jkajaksjjaaaaaaa
2: jkajaksjjaaaaaaa
kdsjkjaskjajsja
aaaaaa
3: aaaaaa
```

Intervalo

5 minutos

Variáveis Indexadas Multi-dimensionais

* São variáveis indexadas de variáveis indexadas, i.e. que cada elemento é uma variável indexada.
* Uso mais comum: variáveis indexadas bi-dimensionais

```
tipo  nome[NumerodeLinhas][NumerodeColunas];
```

```
int a[3][4];
```

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

Inicialiação

```
int a[3][4]={{1,4,5,6}, {3,4,2,1}};
```

Ou:

```
int a[3][4]={1,4,5,6,3,4,2,1};
```

Ou:

```
int main() {
  int a[3][4], i,j;
  for(i=0;i<3;i++)
    for(j=0;j<4;j++)
      scanf("%d",&a[i][j]);
}
```

a	0	1	2	3
0	1	4	5	6
1	3	4	2	1
2	0	0	0	0

* Internamente na memória as variáveis são alocadas em posições de memória consecutivas:

* a[0]
* a[1]
* a[2]

a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[2][0]
a[2][1]
a[2][2]
a[2][3]

* Considerando a variável seguinte, determinar o dia do ano, dado o ano, o mês e dia do mês.

```
int tab_ano[2][13]={
            {0,31,28,31,30,31,30,31,31,30,31,30,31},
            {0,31,29,31,30,31,30,31,31,30,31,30,31}};


int dia_do_ano(int ano, int mes, int dia) {
  int i, bissexto;
  bissexto= ano%4 ==0 && ano%100 !=0 || ano%400==0;
  for(i=1; i<mes;i++)
    dia +=tab_ano[bissexto][i]
  return dia;
}
```

Exercicio: considera o problema inverso:imprima o dia do mês e o mês dado o ano e o dia do ano.

Variáveis bi-dimensionais como argumentos de funções

* Como no caso das variáveis unidimensionais, quando as variáveis bi-dimensionais são passadas como argumentos de função apenas é criada uma nova referência para a mesma variável (a mesma área de memória).
* Na definição da função no parâmetro correspondente é necessário indicar o tamanho do segunda dimensão (que se costuma associar ao número de colunas).
* Isto, porque as posições de memória são reservadas sequencialmente e é necessário saber quando se "muda" de linha.

```
void escrevetab(int a[][10],int n,int m) {
int i,j;
  for(i = 0; i < n; i++) {
    for(j = 0; j < m; j++)
        printf("%d ",a[i][j]);
  printf("\n");
  }
}
```

Relações Binárias

* Dado um conjunto A com N elementos podemos representação de relações binárias $R \subseteq A \times A$ por uma matriz $M_{n \times n}$ tal que:

$M_{ij}=1$ se $(i,j) \in R$

$M_{ij}=0$ se $(i,j) \notin R$

* Escrever funções que determinem se R é reflexiva, simétrica ou transitiva.

```c
#define N 3

int reflexiva(int m[][N],int n) {
  int i = 0;
  while(i < n && m[i][i]) i++;
  if (i >= n) return 1;
  return 0;
}

int simetrica(int m[][N],int n) {
  int i, j;
  for(i = 0 ; i < n ; i++)
    for(j = 0 ; j < i ; j++)
      if(m[i][j] && !m[j][i]) return 0;
  return 1;
}


int transitiva(int m[][N],int n) {
  int i,j,k;
  for(i = 0 ; i < n ; i++)
    for(j = 0 ; j < n ; j++)
      for(k = 0 ; k < n ; k++)
       if(m[i][j]&& m[j][k] &&!m[i][k] ) return 0;
  return 1;
}

main(){
  int m[N][N] = {{1,1,0},{1,1,1},{0,1,0}};
  printf("Reflexiva: %d\n",reflexiva(m,N));
  printf("Simetrica: %d\n",simetrica(m,N));
  printf("Transitiva: %d\n",transitiva(m,N));
}
```

Ordenar uma sequência de linhas de texto lidas, por inserção.

* **Método da Inserção**

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

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

Algoritmo

Guardar as linhas numa variável bi-dimensional `char texto[20][80]` e o usar pesquisa linear.

Ler uma linha de texto, colocá-la em `texto[0]`

Enquanto houver linhas:

* ler uma linha de texto, e colocá-la numa variável auxiliar `linha_lida`
* comparar a linha lida com as anteriormente lidas (por ordem inversa):

```
 se strcmp(linha_lida,texto[j]) < 0 fazer strcpy(texto[j+1],texto[j]) (j=i-1,..0)
```

* até encontrar uma linha "maior" ou chegar ao início do texto. Então, inserir a linha lida na posição:

```
strcpy(texto[j+1],linha_lida).
```

```c
#include <stdio.h>
#define NL 20 /* numero maximo de linhas */
#define LC 80 /* numero  maximo de caracteres por linha */

int main() {
  int j, i, fim=0;
  char linha_lida[LC];
  char texto[NL][LC];
  if (gets(linha_lida)!=NULL)  strcpy(texto[fim],linha_lida);
  fim++;
  while(gets(linha_lida)!=NULL) {
    for(j=fim-1; j>=0 && (strncmp(linha_lida,texto[j],LC) <0) ;
              j--)
       strcpy(texto[j+1],texto[j]);
    strcpy(texto[j+1],linha_lida);
    fim++;
  }
  for (j=0;j<fim;j++) puts(texto[j]);
  return 0;
}
```

Execução:

```
$gcc ordlin.c -ordlin
$cat > teste
ola mundo
ola
olmeiro
```

```
$ ordlin < teste
ola
ola mundo
olmeiro
```

Pesquisa e Classificação num conjunto de dados, representado por uma variável bi-dimensional

* Tabela de valores da precipitação diária em N estações meteorológicas em l /m2

```
int prec[366][100]
```

onde `prec[i][j]` :

precipitação no dia `i` na estação `j`

Esquema do Programa:

1. Introdução de dados
2. Questões (exemplos):
   * Qual a **precipitação anual** na estação 5? E de cada estação?
   * Onde **choveu mais** em 8/11? E no ano?
   * Qual o **dia em que choveu mais** no local 7?
   * Qual o local com **mais dias sem chuva** no ano?
   * Para um dado local, qual a **maior sequência de dias sem chuva**?
   * Supondo que os valores da precipitação anual variam de 0 a 100, classificar os dados em 10 classes e determinar a frequência absoluta de cada classe.

1. Dados : gerar aleatoriamente valores de teste.

```
#define N 10
#define DIAS 366
#define MAX_PREC_DIA 10.0
gera_dados() {
  int i,j;
  srand(time(NULL));
  for(i = 1; i < DIAS; i++)
    for(j = 1; j < N; j++) prec[i][j]= rand()%MAX_PREC_DIA);
}
```

```
void escreve_dados (void) {
  int i,j;
  printf(" Est:   ");
  for(j = 1; j < N; j++) printf("%2d ",j);
  printf("\n-----------------------------------\n");
  for(i = 1; i < DIAS; i++) {
    dia_e_mes(i); putchar(' ');
    for(j = 1; j < N; j++) printf("%2d ",prec[i][j]);
    putchar('\n');}}
```

Início da tabela

Est:	1	2	3	4	5	6	7	8	9	10
1/ 1	5	6	3	2	6	8	5	5	5	6
2/ 1	7	8	5	1	7	5	0	1	5	5
3/ 1	1	3	6	7	0	5	2	3	1	5
4/ 1	6	6	3	2	0	9	0	6	6	7
5/ 1	4	5	7	9	8	4	6	8	5	3
6/ 1	4	6	8	2	4	9	8	6	4	1
7/ 1	3	1	9	8	5	9	7	7	7	6
8/ 1	5	1	3	2	2	2	7	0	0	4
9/ 1	6	6	1	4	9	7	5	7	5	0
10/ 1	8	1	3	7	9	8	8	9	7	6
11/ 1	5	2	9	8	7	4	2	6	4	3
12/ 1	0	0	9	3	7	8	0	2	7	8
13/ 1	2	7	9	7	6	8	5	5	7	5
14/ 1	3	4	7	2	5	6	6	7	2	3
15/ 1	0	3	5	2	6	2	2	9	5	2
16/ 1	7	9	9	6	7	8	6	4	5	4
17/ 1	9	8	0	9	2	5	5	1	3	0

2. Programa principal

```
char questoes[][80]={
  "1- Qual a precipitacao anual na estacao 5?\n",
  "2- Qual o local onde choveu mais no dia 8 de Novembro?\n",
  "3- Qual o dia em que choveu mais no local 7?\n",
  "4- Qual o local com mais dias sem chuva durante o ano?\n",
  "5- Para um dado local, qual a maior sequencia de dias sem chuva?\n",
  "6- Classificacao\n"};
int prec_anual(int);
void prec_at(void);
int max_linha(int);
int max_anual(void);
int zeros(void);
int seca(int);
void classifica(void);
int main() {
  int i=0;
  gera_dados();
  printf("%s %d\n",questoes[i++], prec_anual(5));
  printf("%s %d\n",questoes[i++], max_linha(dia_do_ano(8,11)));
  printf("%s %d\n",questoes[i++], max_anual());
  printf("%s %d\n",questoes[i++], zeros());
  printf("%s %d\n",questoes[i++],seca(4));
  prec_at(); /* calcula precipitacoes anuais */
  printf("%s\n",questoes[i++]);
  classifica();
  return 0}
```

1. Qual a precipitação anual na estação 5? E de cada estação?

Precipitacão anual na estação `j`: somar os valores de `prec[i][j]` tendo `j` fixo e variando `i` de `1` a `365 (ou 366)`:

```
int prec_anual(int e){
   int i, s=0;
    for(i=1;i<DIAS;i++) s+=prec[i][e];
    return s;
 }
```

Precipitacões em todas as estações:

```
void prec_at(void) {
  int i;
  for(i=1;i<N;i++) prec_total[i]= prec_anual(i);
}
```

2. Onde choveu mais no dia 8 de Novembro?

Dado um dia `d` calcular o máximo da "linha":

```
int max_linha(int d) {
  int j, max,k;
  max = prec[d][1];
  for(k = 1,j = 2;j < N;j++)
    if (prec[d][j] > max) {k = j; max = prec[d][j];}
  return k;
}
```

e qual a precipitacão máxima durante o ano:

```
int max_anual() {
    int dmax = 1, emax = 1, i,j;
    for(i = 1; i < DIAS; i++)
      for(j = 1; j < N; j++)
        if (prec[i][j] > prec[dmax][emax]){
            dmax = i; emax = j;}
    return prec[dmax][emax];
  }
```

4. Qual local com mais dias sem chuva no ano? Contar os zeros para cada local e depois determinar o máximo desses valores.

```
int secos[N]={0};
int zeros() {
  int j, i;
  for(j = 1; j < N; j++)
    for(i = 1; i < DIAS; i++) if(!prec[i][j]) secos[j]++;
    return (max(secos,N));
}

int max(int v[],int n) {
  int j,imax;
  for(imax = 1,j = 2 ; j < n ; j++)
      if(v[j] > v[imax]) imax = j;
  return imax;
}
```

5. Para cada estação procurar a maior subsequência com zeros.

```
int seca(int e) {
  int cmax = 0, imax = 0, i = 1;
  int c, inic;
  while(i < DIAS) {
    c = 0; inic = i;
    while(i < DIAS && !prec[i][e]) {c++; i++;}
    if (c > cmax) {cmax = c; imax = inic;}
    i++; }
  return cmax;
}
```

6. Classificação: `classe[i]=` número de locais com prec. entre `i*10` e `(i+1)*10-1`

```
void classifica(void){
  int classe[10], i,c;
  for(i = 0; i < 10; i++) classe[i] = 0;
  for(i = 1;i < N;i++) {
    c = prec_total[i]/10;
    classe[c]++;
  }
  for(i = 0;i < 10;i++)
      printf("%4d-%4d%2u\n",i*10,(i+1)*10-1,classe[i]);
}
```