UAlg FCT

UNIVERSIDADE DO ALGARVE

FACULDADE DE CIÊNCIAS E TECNOLOGIA

Laboratório de Programação

Lição n.º 2

Busca e ordenação com cadeias de caracteres

Busca e ordenação com cadeias de caracteres

- Busca em arrays de cadeias.
- Ordenação de arrays de cadeias.

Busca linear

- A busca linear num array de cadeias é análoga à busca linear num array de int:

```
int strings_find(char **s, int n, const char *x)
{
  for (int i = 0; i < n; i++)
    if (strcmp(s[i], x) == 0)
      return i;
  return -1;
}
```

- Mas atenção: a comparação faz-se com a função de biblioteca strcmp.

strcmp

- A função strcmp compara duas cadeias de carateres usando a ordem *lexicográfica* dos valores numéricos dos bytes.
- Dá zero, se as duas cadeiras forem iguais, caráter a caráter.
- Dá um número negativo indeterminado se a cadeia no primeiro argumento for lexicograficamente menor que a cadeia no segundo argumento.
- Dá um número positivo indeterminado se a cadeia no primeiro argumento for lexicograficamente maior que a cadeia no segundo argumento.

Se as cadeias em análise só tiverem letras minúsculas ou só letras maiúsculas, sem acentos ou outros sinais diacríticos, a ordem lexicográfica coincide com a ordem alfabética habitual.

Comparação de cadeias

- Para comparar cadeias, não se usa <, <=, ==, >, >= ou !=.
- Usa-se strcmp.
- Por exemplo:
  - strcmp(s, t) < 0 significa "s menor que t".
  - strcmp(s, t) == 0 significa "s igual a t".
  - strcmp(s, t) != 0 significa "s diferente de t".
  - strcmp(s, t) <= 0 significa "s menor ou igual a t".
  - strcmp(s, t) > 0 significa "s maior que t".
  - strcmp(s, t) >= 0 significa "s maior ou igual a t".

Teste unitário para strcmp

- Observe:

```
void unit_test_strcmp(void)
{
  assert(strcmp("lisboa", "faro") > 0);
  assert(strcmp("quarteira", "queluz") < 0);
  assert(strcmp("tavira", "Tavira") != 0);
  assert(strcmp("lagos", "LAGOS") != 0);
  assert(strcmp("silves", "silves") == 0);
  assert(strcmp("braga", "braganca") < 0);
  assert(strcmp("vila real de santo antonio",
                "vila real") > 0);
  assert(strcmp("", "sagres") < 0);
  assert(strcmp("alcoutim", "") > 0);
  assert(strcmp("", "") == 0);
}
```

Exemplo: consultando as cidades

- Eis uma função que lê para um array um ficheiro com a lista das cidades portuguesas e depois interroga esse array para cada nome lido da consola:

```
void test_portuguese_cities(void)
{
  FILE *f = fopen("cidades_2011.txt", "r");
  char *cities[MAX_CITIES];
  int n = strings_read(f, cities);
  char line [MAX_LINE_LENGTH];
  while (str_getline(line) != EOF)
  {
    int k = strings_find(cities, n, line);
    printf("%d\n", k);
  }
}
```

```
$ ./a.out
faro
43
gambelas
-1
aveiro
16
estoi
-1
tavira
134
```

Busca dicotómica

- Adaptamos as funções usadas com arrays de int:

```
int strings_rank(char **s, int n, const char *x)
{
  int result = 0;
  while (n > 0)
  {
    int m = n / 2;
    if (strcmp(x, s[m]) <= 0)
      n = m;
    else
    {
      result += m+1;
      s += m+1;
      n -= m+1;
    }
  }
  return result;
}

int strings_bfind(char **s, int n, const char *x)
{
  int r = strings_rank(s, n, x);
  return r < n && strcmp(s[r], x) == 0 ? r : -1;
}
```

Exemplo: consultando palavras inglesas

- Eis uma função que consulta dicotomicamente uma lista ordenada de 10000 palavras inglesas, carregadas num array de cadeias:

```
void test_english_words(void)
{
  FILE *f = fopen("wordlist.10000.txt", "r");
  char *words[MAX_WORDS];
  int n = strings_read(f, words);
  char line [MAX_LINE_LENGTH];
  while (str_getline(line) != EOF)
  {
    int k = strings_bfind(words, n, line);
    printf("%d\n", k);
  }
}
```

```
$ ./a.out
house
4291
batata
-1
potato
6829
gato
-1
cat
1397
Program
6998
```

Ordenação de arrays de cadeias

- Atenção: ao ordenar arrays de cadeias dinâmicas, as cadeiras não mexem, no heap.
- O que mexe, por troca, são os apontadores no array de apontadores.
- Para trocar dois apontadores num array de apontadores para char, usamos a seguinte função:

```
void strings_exchange(char **a, int x, int y)
{
  char *m = a[x];
  a[x] = a[y];
  a[y] = m;
}
```

Note bem: ao fazer a[x] = a[y] não estamos a copiar a cadeia a[y] para a cadeia cadeia a[x]. Estamos sim a copiar o endereço existente na posição y do array a para a posição x do mesmo array.

Insertionsort para array de cadeias

- Adaptamos a função ints_isort:

```
void strings_sort_last(char **a, int n)
{
  int i = n-1;
  while (i > 0 && strcmp(a[i-1], a[i]) > 0)
  {
    strings_exchange(a, i-1, i);
    i--;
  }
}
```

```
void strings_isort(char **a, int n)
{
  for (int i = 2; i <= n; i++)
    strings_sort_last(a, i);
}
```

Testando o insersionsort com cadeias

- Ler algumas palavras e ordenar, repetidamente:

```
void test_strings_isort_demo(void)
{
  char *a[1000];
  int n;
  while ((n = strings_getwords(a)) != 0)
  {
    strings_isort(a, n);
    strings_println(a, n, " ");
    freopen("/dev/tty", "r", stdin);  // Unix
//  freopen("CON", "r", stdin);       // Windows
    printf("------\n");
  }
}
```

```
$ ./a.out
sardinha carapau bacalhau pescada dourada cherne faneca
bacalhau carapau cherne dourada faneca pescada sardinha
------
morango uva laranja banana quivi
clementina manga ameixa framboesa
ameixa banana clementina framboesa laranja manga morango quivi uva
------
```