Laboratorio di Algoritmi e Strutture Dati

Esercitazioni del'11 Dicembre 2012

Esercizio 4: strtok

La libreria standard contiene la funzione `strtok` il cui prototipo

```
char *strtok(char *s1, const char *s2);
```

è definito in `string.h`.
La funzione `strtok` suddivide la stringa `s1` in *parole* (*token*), vale a dire sequenze consecutive di caratteri separate da uno o più caratteri fra quelli che appaiono nella stringa `s2`.
`strtok` funziona in questo modo:

- Supponiamo di voler *tokenizzare* la stringa

  ```
  pippo, pluto e topolino
  ```

  memorizzata in un array `str`, rispetto ai caratteri di separazione: spazio `' '`, tabulazione `'\t'`, newline `'\n'`, carriage return `'\r'`, virgola `','`, e `'e'`.

- Allora, ponendo i caratteri di separazione in una stringa

  ```
  char *sep = " \t\n\r,e";
  ```

  e dichiarando una variabile `char *t`, la prima chiamata a `strtok`, deve essere la seguente:

  ```
  t = strtok(str,sep);
  ```

- Dopo la prima chiamata, `t` punta al (primo carattere del) primo token di `str`, nell'esempio `"pippo"`.
- Le successive chiamate a `strtok` devono passare `NULL` come primo argomento:

  ```
  t = strtok(NULL,sep);
  ```

  dopo ogni chiamata `t` punta al token successivo di `str`: prima `"pluto"`, poi `"topolino"`.

- Quando la stringa è stata completamente tokenizzata, la chiamata:

  ```
  t = strtok(NULL,sep);
  ```

  pone `t` a `NULL`.

Si richiede di implementare una funzione `tokenizza`, di prototipo

```
char *tokenizza(char *s1, const char *s2);
```

che si comporti come `strtok`. In particolare, si consideri il programma contenente la seguente funzione `main`:

```
int main(void)
{
    char line[MAXDIM+1];
    char sep[] = " \n\t\r";
    char *t = line;

    printf("Immetti linea:\n");
    while((*t++ = getchar()) != '\n');
    *--t = 0;
    t = tokenizza(line,sep);
    while(t) {
        printf("[%s]\n",t);
        t = tokenizza(NULL,sep);
    }
    putchar('\n');

    return 0;
}
```

e in esecuzione si specifichi in input la stringa:

```
pippo pluto e topolino
```

Il programma dovrà produrre il seguente output:

```
[pippo]
[pluto]
[e]
[topolino]
```

Si osservi che `tokenizza` può operare in questo modo:

1. Alla prima chiamata `tokenizza` inizia a scandire `line` dal suo primo carattere fino a trovare il primo carattere di separazione, diciamo in posizione di indice `i`.
2. A questo punto rimpiazza il carattere in `line[i]` con il carattere `'\0'` di terminazione della stringa, e restituisce al chiamante l'indirizzo del primo carattere di `line`.
3. IMPORTANTE: prima di terminare l'esecuzione, memorizza in una variabile `static` l'indirizzo di `line[i+1]`.
4. La successiva chiamata a `tokenizza`, il cui primo argomento è `NULL`, inizia a scandire `line` dall'indirizzo di `line[i+1]`, precedentemente memorizzato, e poi procede come ai punti 1), 2) e 3), ponendo il carattere di terminazione al posto del primo separatore incontrato, e memorizzando l'indirizzo successivo a questo carattere.
5. `tokenizza` procede come nel punto 4) nelle chiamate successive, fino a incontrare la terminazione della stringa `line`.