Dicas para a primeira etapa

LA1/LI2

Aula 2
1 de Março de 2015

A função `scanf` e a sua variante para strings `sscanf` pode ser utilizada para desmembrar strings. Seguem-se alguns formatos mais comuns:

espaço	ignorar whitespace que possa existir
%d	int
%u	unsigned int
%f	float
%lf	double
%s	string não vazia sem *whitespace*
%[	string não vazia que tem que respeitar a expressão regular dada entre [e]; não ignora *whitespace* antes de ler

Nos casos das strings recomenda-se a utilização de um tamanho máximo do campo.

Exemplos de formatos de scanf

`"%s %s"`	duas palavras; cuidado com buffer overrun!
`"%20s %10s"`	palavra com no máximo 20 caracteres seguida de outra com no máximo 10
`"%[^\n]"`	string não vazia (tudo excepto `\n`)
`"%[aeiou]"`	string não vazia composta únicamente por vogais
`"%[a-z]"`	string não vazia composta por letras minúsculas
`"%[a-zA-Z]"`	string não vazia composta por letras
`"a%[^\n]"`	'a' seguido de uma string não vazia
`"a%[^b]b%s"`	'a' seguido de uma string não vazia que não pode conter 'b' seguido de uma palavra

Truques com o scanf

Lembre-se que o scanf devolve o nº de argumentos lidos. A conjunção disso com formatos permite que o sscanf seja muito poderoso a tratar strings.

- Função que permite desmembrar uma string
- Usa um separador para separar a string em tokens
- A função tem dois argumentos, sendo o segundo uma string com os separadores possíveis
- O separador tem que ser um caractere
- Da primeira vez o primeiro argumento é a string que se quer separar
- Das vezes seguintes o primeiro argumento é NULL
- Devolve NULL quando não há mais tokens

Exemplos do strtok

```
/**
Separa uma linha por palavras e
imprime cada palavra numa linha
@param linha Uma string contendo uma linha
*/
void imprimir_palavras(char *linha) {
    char *separador = " \t\n\r";
    char *pal = NULL;

    pal = strtok(linha, separador);
    printf("%s\n", pal);

    while((pal = strtok(NULL, separador)) != NULL) {
        printf("%s\n", pal);
    }
}
```

ctype Testa o tipo de um caractere (e.g., isalpha, isalnum, isblank, isspace, islower, isupper)

string Funções úteis sobre strings (e.g., strcpy, strcat, strcmp, strchr, strstr, strtok)

```c
#define MAX_TAM 10
/* Array (primeiro indice 0, ultimo MAX_TAM - 1) */
int valor[MAX_TAM];

/* Matriz quadrada */
char matriz[MAX_TAM][MAX_TAM];

/* Colocar um valor no array */
valor[0] = 2;

/* Colocar um valor na matriz */
matriz[2][3] = 1;
```

```c
/* Definir um novo tipo de dados */
typedef struct coord {
    int x;
    int y;
} coord;

/* Declaracao de uma variavel desse tipo */
coord c;
c.x = 1;
c.y = 2;

/* Declaracao de um array desse tipo */
coord casas_usadas[MAX_TAM];
```

Dicas de programação

- Escrever funções pequenas;
- Não usar copy & paste;
- Criar funções auxiliares;
- Dar nomes sugestivos às funções e variáveis;
- Não aceder directamente às estruturas de dados;
- Se se pretender mudar a estrutura de dados é preciso modificar todos os sítios no código onde esta é acedida;
- Usar macros ou funções auxiliares para o fazer.

```
/* Estrutura */
typedef struct tabuleiro {
    char tab[100][100];
    int lins;
    int cols;
} tabuleiro;
/* Definicao das macros */
#define COLUNAS(est)      est.cols;
#define LINHAS(est)       est.lins;
#define POS(est, x, y)    est.tab[y][x]
/* Utilização */
tabuleiro estado;

printf("%d\n", LINHAS(estado));
```