LABORATÓRIOS DE INFORMÁTICA III

2013/2014
LEI

2º ANO - 2º SEM

Aula Comum: Projecto de C

F. Mário Martins (fmm@di.uminho.pt)
João Luís Sobral (jls@di.uminho.pt)

DI/UM

CALENDÁRIO

CALENDÁRIO LI3 - Versão 1
2013-2014

Semana	2a.feira	3a.feira	4a. Feira	5a.feira	6a.feira	sábado	
17/02 a 22/02							**Semana de LEI**
24/02 a 01/03			COMUM				**Aula comum de apresentação de LI3 (TP de C)**
03/03 a 08/03							
10/03 a 15/03							
17/03 a 22/03			COMUM				
24/03 a 29/03							
31/03 a 05/04							
07/04 a 12/04							
14/04 a 19/04							
21/04 a 26/04			COMUM				**TP de Java + Entrega electrónica do TP de C**
28/04 a 03/05							**Avaliações do TP de C**
05/05 a 10/05							
12/05 a 17/05							Semana da queima
19/05 a 24/05			COMUM				
26/05 a 31/05							
02/06 a 07/06							
09/06 a 14/06							**Entrega electrónica do TP de Java**
16/06 a 21/06							**Avaliações do TP de Java**
23/06 a 28/06							**Lançamento das Notas Finais**
30/06 a 05/07							
07/07 a 12/07							

	Aula comum em sala a marcar
	entregas electrónicas de trabalhos
	avaliações presenciais dos trabalhos
	laboratórios de C
	laboratórios de Java
	férias feriados
	queima
	Notas finais da UC

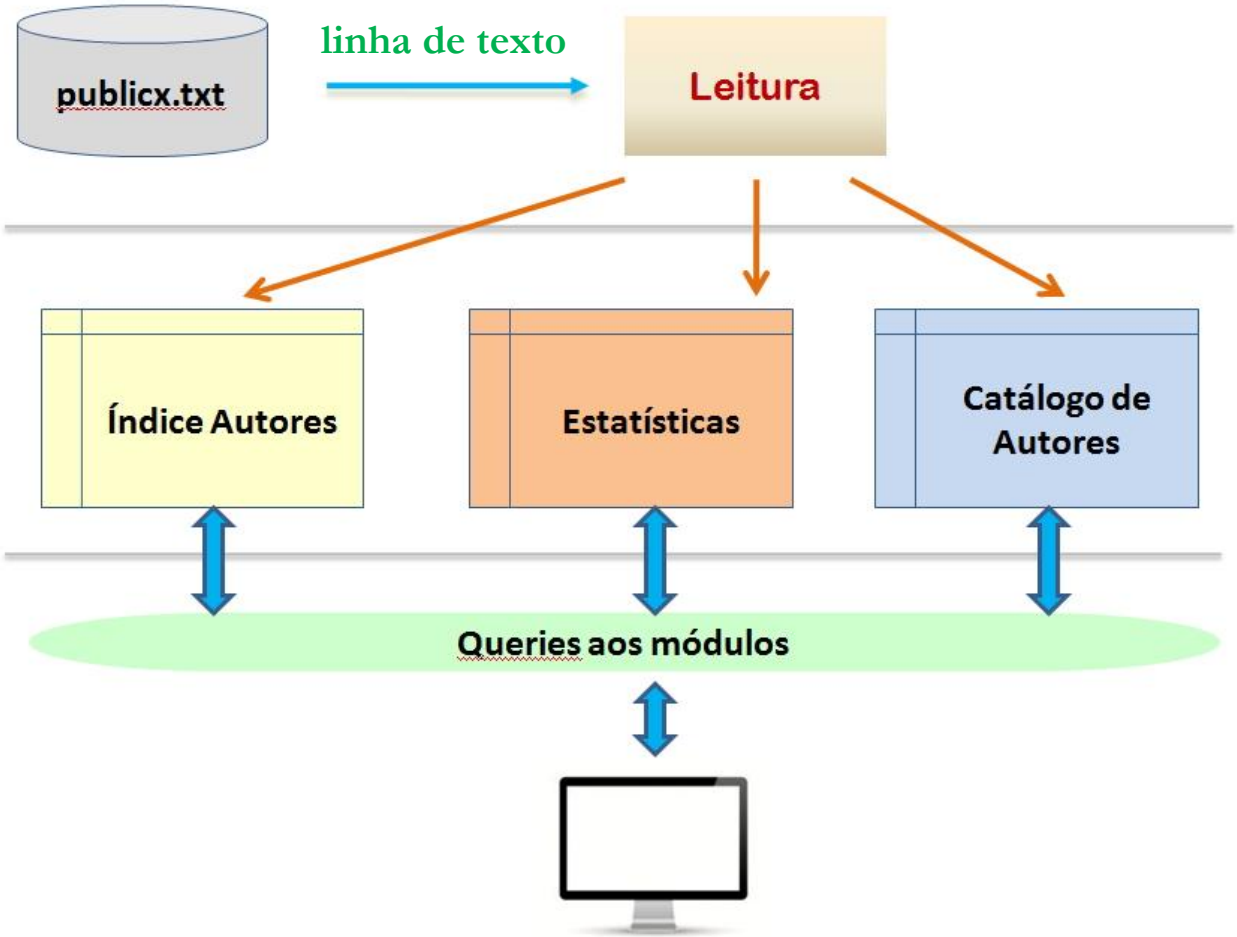
LABORATÓRIOS DE INFORMÁTICA III

2013/2014
LEI

TRABALHO PRÁTICO DE C

- ☑ Observações gerais;
- ☑ Questões de modularidade;
- ☑ Estruturas de dados;

Arquitectura da aplicação



Projecto de C: GESTAUTS

✍ Importância de realizar algum profiling quando se desenvolvem aplicações de grande escala, em especial quanto aos dados; Como fazer? Testar os dados e fazer algumas medidas;

☑ Exemplos:

- nº de linhas em puclix.txt → 136127
- buffer para fgets(s, ?, fich) → 1024
- char[?] para o nome de cada autor → 200
- nº de autores por publicação → 99% - 1 a 10
- limite razoável para os anos das publicações → 50
- total de nomes de autores → 368014
- média de nomes de autores por letra aceitável → +- 5000

Projecto de C: GESTAUTS

✍ Importância de ter uma abordagem ao projecto segura e por fases;

☑ FASE 1: Garantir que cada linha é lida correctamente do ficheiro e, em seguida, garantir que dela se extraem correctamente os nomes dos autores e o ano (sem pensar ainda nos módulos) =>

- fgets(linha, 1024, fich) correcto (atenção a \n\0 em linha);
- strtok() da linha correcto (n strings + 1 int em memória);
- nomes devem ser limpos de espaços iniciais e finais => trim() ;
- atenção no uso da função atoi();
- testar a funcionalidade fazendo printf() dos tokens extraídos. São estes que vão ser inseridos nos módulos e se estiverem errados tudo estará errado !

Projecto de C: GESTAUTS

Importância de ter conhecimentos seguros sobre coisas básicas;

- Existe alguma confusão entre char [], char* e o conceito de "string" em C;
- Uma "string" é o prefixo de um char [] **"terminado" por** '\0' (delimitador obrigatório);
- Um char [] pode conter várias "strings" !
- char* strcpy(char [], char*) char* strcat(char* , char*) char* strdup(char*)

```
char *
strcpy(char *s1, const char *s2)
{
        char *s = s1;
        while ((*s++ = *s2++) != 0)
                ;
        return (s1);
}
```

(Não aloca espaço)

```
char *
strdup(const char *str)
{
 size_t siz;
 char *copy;

 siz = strlen(str) + 1;
 if ((copy = malloc(siz)) == NULL)
         return(NULL);
 (void)memcpy(copy, str, siz);
 return(copy);
}
```

(Aloca espaço e copia)

- Atenção: malloc() não inicializa; memset() não aloca; calloc() = malloc() + memset() ;
- Má compreensão destes mecanismos => segmentation faults;
- Não descurar nunca o uso de memcopy() e realloc(); Nunca usar bcopy();

Projecto de C: GESTAUTS

Importância de ter conhecimentos seguros sobre coisas básicas;

strncpy(3) - Linux man page

Name

strcpy, strncpy - copy a string

Synopsis

```
#include <string.h>

char *strcpy(char *dest, const char *src);

char *strncpy(char *dest, const char *src, size_t n);
```

Description

The **strcpy**() function copies the string pointed to by *src*, including the terminating null byte ('\0'), to the buffer pointed to by *dest*. The strings may not overlap, and the destination string *dest* must be large enough to receive the copy. *Beware of buffer overruns!* (See BUGS.)

The **strncpy**() function is similar, except that at most *n* bytes of *src* are copied. **Warning**: If there is no null byte among the first *n* bytes of *src*, the string placed in *dest* will not be null-terminated.

If the length of *src* is less than *n*, **strncpy**() writes additional null bytes to *dest* to ensure that a total of *n* bytes are written.

A simple implementation of **strncpy**() might be:

```
char *
strncpy(char *dest, const char *src, size_t n)
{
    size_t i;

   for (i = 0; i < n && src[i] != '\0'; i++)
        dest[i] = src[i];
    for ( ; i < n; i++)
        dest[i] = '\0';

   return dest;
}
```

Return Value

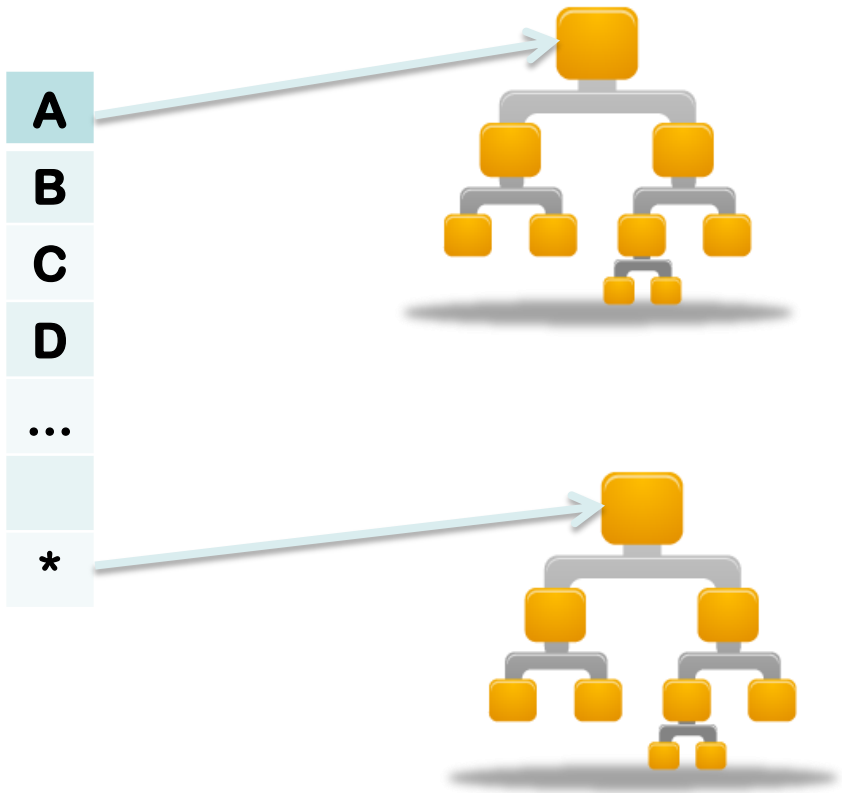
The **strcpy**() and **strncpy**() functions return a pointer to the destination string *dest*.

Pelos vistos strncpy() está na moda.
Porquê ?

Projecto de C: GESTAUTS

☑ **FASE 2.1: Pensar na arquitectura de cada módulo em função das queries requisitadas pelo projecto. Pensar na estrutura de dados interna ao módulo e pensar nas funções que serão disponibilizadas aos clientes dos módulos (o cliente principal será o main()).**

EXEMPLO:



Projecto de C: GESTAUTS

☑ FASE 2.2: Declarações .h

BinarySearchTree.h

```
typedef int ElementType;

#ifndef _BINARY_SEARCH_TREE_H
#define _BINARY_SEARCH_TREE_H
struct TreeNode;
typedef struct TreeNode* Position;
typedef struct TreeNode* SearchTree;

SearchTree MakeEmpty(SearchTree T);
Position Find(ElementType X,SearchTree T);
Position FindMin(SearchTree T);
Position FindMax(SearchTree T);
SearchTree Insert(ElementType X,SearchTree T);
SearchTree Delete(ElementType X,SearchTree T);
ElementType Retrieve(Position P);
#endif
```

```
1  #ifndef AVLTREE_H_INCLUDED
2  #define AVLTREE_H_INCLUDED
3
4  typedef struct node
5  {
6      int data;
7      struct node*  left;
8      struct node*  right;
9      int      height;
10 } node;
11
12
13 void dispose(node* t);
14 node* find( int e, node *t );
15 node* find_min( node *t );
16 node* find_max( node *t );
17 node* insert( int data, node *t );
18 node* delete( int data, node *t );
19 void display_avl(node* t);
20 int get( node* n );
21 #endif // AVLTREE_H_INCLUDED
```

A qualidade e extensão de uma API (.h em C) é muito relevante para a utilidade do módulo desenvolvido.

Projecto de C: GESTAUTS

Podem encontrar-se na web imensas implementações das estruturas de dados que serão candidatas a soluções para este projecto; Há que ter no entanto em consideração o seguinte:

1) Há imensas implementações de muitos tipos de árvores, de muitos tipos de maps (tabelas de hashing), de grafos, etc., mas nem todas são correctas e apenas algumas são verdadeiramente certificadas.

```
                    GNU LESSER GENERAL PUBLIC LICENSE
                        Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.
```

2) Reutilizar software é boa prática. Se correcto melhor. Se licenciado e livre ainda melhor, em especial quando se refere a fonte;

3) As estruturas de dados de base (árvores, tabelas de hashing, etc.) não terão neste projecto qualquer valorização. O que será avaliado qualitativamente será a razão da sua escolha, as adaptações feitas, o seu dimensionamento e a sua utilização final na solução.

ENCAPSULAMENTO EM C

Assim, em C, o encapsulamento pode ser garantido se as variáveis forem declaradas static tal como sugerido e aconselhado em manuais de C.

Static storage class designation can also be applied to external variables. The only difference is that static external variables can be accessed as external variables only in the file in which they are defined. No other source file can access static external variables that are defined in another file.

```
/* File: xxx.c */
     static int count;
     static char name[8];
     main()
     {
          ... /* program body */
     }
```

Variáveis static external

- Ler documento sobre Modularidade em C (pasta de Conteúdos do BB) para mais informações importantes.