

# Arquitetura de Computadores 2008/09

## Aula prática 3 – Apontadores em C

1. Considere o seguinte programa:

```
#include <stdio.h>
int main()
{
    int i1 = -1;
    int i2 = -2;
    int i3 = -3;

    int *pi1 = &i1;
    int *pi2 = &i2;
    int *pi3 = &i3;

    *pi1 = -4;
    *pi2 = *pi1;

    pi2 = pi1;
    pi3 = pi2;

    *pi3 = -5;

    printf("%d\n%d\n%d\n", i1, i2, i3);
    return 0;
}
```

Diga quais os resultados que este programa imprime no ecrã. Confirme a sua resposta correndo o programa. Se a sua resposta não estiver certa tente perceber porquê.

2. Programe a função **mystrlen**, que devolve o tamanho de uma *string*, utilizando o incremento de apontadores, da forma mais abreviada possível. Faça um programa para testar a função, em que pede uma *string* ao utilizador e apresenta o seu comprimento.

3. Programe a função:

```
void mymemcpy( char *m1, char *m2, int n )
```

A função copia *n* bytes a partir do endereço *m2* para a zona de memória que inicia no endereço *m1*, utilizando apontadores e o operador de desreferenciação da forma mais abreviada possível.

Diga por que é que o argumento *n* é necessário se *m1* e *m2* forem vectores (porque não usar **sizeof**?).

Veja o manual da função **memcpy** (man memcpy).

4. Programe a função:

```
void s_replace( char s[], int n, char rep[] )
```

A função substitui os caracteres de **s** a partir da posição **n** pelos da *string* **rep**. Utilize para isso a função **memcpy** programada no exercício anterior. Assuma que **rep** cabe dentro de **s**, i.e., a *string* resultante continua com o mesmo tamanho de **s**.

Faça um programa para testar a função, em que pede **s**, **n** e **rep** e faz a substituição.

Exemplo:

```
Introduza s: impossivel
Introduza n: 2
Introduza rep: @@@
Resultado: im@@@sivel
```

5. Diga qual a diferença entre **NULL** e **'\0'**. Dê um exemplo da afectação de cada uma destas constantes a uma variável.

6. Considere o seguinte programa:

```
#include <stdio.h>
#include <string.h>
#define MAX_S 100

int main() {
    char s1[MAX_S];
    char *s2;

    s1 = s2;
    s2 = s1;

    strcpy(s2, "abc");
    strcpy(s1, s1);

    s1++;
    s2++;

    (*s1)++;
    (*s2)++;

    printf("%s\n%s\n", s1, s2 );
    return 0;
}
```

Neste programa há duas linhas que dão erro de compilação. Quais são e porquê?

Se apagar essas duas linhas, quais são os resultados que o programa vai apresentar no ecrã?

Verifique as sua respostas compilando e executando o programa.

7. Programe a função:

```
char *mystrdup( char *s )
```

Esta função cria uma cópia da *string* *s* em memória dinâmica e devolve o endereço da cópia, utilizando as funções apresentadas nos acetatos das aulas práticas 1 e 2.

Veja o manual da função **strdup** (`man strdup`).

8. Refaça a função do exercício 7 da aula anterior (que retorna o menor elemento do vector), mas usando apontadores e desreferenciação para aceder aos elementos do vector.

9. Analise e explique o objectivo e comportamento da seguinte função:

```
void dosomething (char *s1, char *s2) {  
    while (*s1++ = *s2++)  
        ;  
}
```