Arquitectura de Computadores 2010/11

Aula prática 4 – Apontadores em C

1. Considere o seguinte programa:

```
#include <stdio.h>

int main()
{

    char c1 = 'a';
    char c2 = 'b';

    char *p1 = &c1;
    char *p2 = &c2;

    *p1 = 'c';
    *p2 = *p1;

    printf("%c\n%c\n", c1, c2 );
    printf("%d\n%d\n", *p1 == *p2, p1 == p2 );
    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. Considere que:

```
int a[2];

a[0] = 2;
a[1] = 0;
```

   Indique o resultado das seguintes expressões em C:

   a. `*a`

   b. `*a+1`

   c. `*( a+1 )`

   d. `(*a) * (*(a+1))`

   e. `a[0] = a[1]`

   Valide as suas respostas experimentalmente com um pequeno programa em C, que apresente no ecrã o resultado das diversas expressões.

3. 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.

4. 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`).

5. 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 **mymemcpy** 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
```

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

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

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