Linguagens e Ambientes de Programação (2018/2019)

Prática 12

Ligações. Escopo estático e escopo dinâmico. Modelo de execução. Exercícios de 40 a 45.

Esta deveria ter sido a aula prática 8 mas, por culpa do calendário muito irregular da altura da Páscoa, teve de ser desviada para a última semana de aulas.

Esta matéria é importante e grande parte dela aparecerá garantidamente no 2º teste de LAP.

- 40 - Considere o seguinte programa em C:

```
#include <stdio.h>
int a;    // variável a-global
int b;    // variável b-global

void p(void) { a = b; }

void f(int b) { // parâmetro b-local
    int a = b; // variável a-local
    p(); /* aqui */
}

int main(void) {
    a = 5; b = 6; f(7);
    printf("%d %d\n", a, b);
    return 0;
}
```

- a) Qual o output deste programa, escrito em C.
 - b) Qual seria o output deste programa se o C utilizasse escopo dinâmico nas chamadas de funções.
-
- 41 - No programa da pergunta 40, diga qual é **ambiente** no ponto indicado pelo comentário: "aqui".
-
- 42 - No programa da pergunta 40, diga qual é o **âmbito** da variável global "b".
-
- 43 - Escreva em C uma função polimorfica para implementar um caso particular de atribuição entre arrays: uma posição dum array recebe o conteúdo de outra posição de outro array. Use a técnica de manipulação direta de memória usando apontadores **void *** e a função **memcpy**, que foi usada na aula teórica 14. Se a cópia tiver sucesso, deve ser retornado o valor **true**. Se a cópia tiver insucesso, por algum dos índices ser inválido, deve ser retornado o valor **false**. Comece por identificar os argumentos e tipos da função e depois escreva o corpo da função. Na função **main** exemplifica-se uma chamada da função pretendida.

```
#include <stdbool.h>
#include <string.h>

bool copy(...)
{
```

```

}

int main(void)
{
    const ASIZE = 10, BSIZE = 20;
    int a[ASIZE], b[BSIZE];
    copy(sizeof(int), a, ASIZE, 3, b, BSIZE, 6); // a[3] = b[6]
    return 0 ;
}

```

- 44 - Mostre o estado da pilha à entrada da função h, durante a execução do seguinte programa GCC. Trata-se do segundo exercício do final da aula teórica 15. É parecido com o primeiro exercício do final da aula teórica 15 (já resolvido), sendo a única diferença a passagem da função h como parâmetro para a função g, o que obriga a usar uma closure.

(Extra: Já agora, veja ainda o que acontece se ainda for execução da instrução dentro da função h.)

```

#include <stdio.h>

int i = 1, j = 1, k = 1, l = 1 ;

void f(void) {
    int i = 2, j = 2 ;
    void h(void) {          // função aninhada
        int i = 3 ;
        l = i + j + k ;
    }
    void g(void p(void)) { // outra função aninhada com parâmetro funcional
        int i = 7, j = 7, k = 7, l = 7 ;
        p() ;
    }
    g(h) ;
}

int main(void)
{
    f() ;
    printf("%d\n", l) ;
    return 0 ;
}

```

- 45 - Mostre o estado da pilha de execução do seguinte programa GCC, no exato momento em que a pilha atinge a sua altura máxima.

(Extra: Já agora, depois continue a execução do programa até ao final.)

```

#include <stdio.h>

int fact(int x) {
    if( x == 0 )
        return 1 ;
    else
        return x * fact (x - 1) ;
}

int main() {
    int a = 3 ;
    int f = fact(a) ;
    printf("%d\n", f) ;
    return 0 ;
}

```

- 46 - Resolva o exercício que aparece no final da aula teórica 17. Confirme que a ferramenta Valgrind descobre todos os erros de execução desse programa e os assinala claramente. Não use o Eclipse, porque a

compilação e a execução terão de ser feitas na linha de comando (na compilação, usar as opções -g -00).

- 47 - A biblioteca "crypt", referida na teórica 17, permite adicionar aos programas suporte para passwords codificadas. Nos exercícios seguintes, use o código da página:

http://www.gnu.org/software/libc/manual/html_node/crypt.html

- a) Copie, compile na linha de comando (com a opção -lcrypt), e corra o primeiro programa. Ao correr o programa, introduza uma password P do seu agrado e guarde a string S escrita pelo programa. Trata-se duma string S que representa P, mas a partir da qual é virtualmente impossível descobrir qual é esse P. A string S pode até ser tornada pública sem comprometer a segurança de P.
 - b) Copie o segundo programa, trocando o valor de inicialização da variável pass pela string S gerada pelo programa anterior. Agora compile (com a opção -lcrypt) e corra o segundo programa. Verificará que o programa consegue detetar quando é introduzida a password correta P. Mas a password P continua secreta e só o dono a conhece.
-
-