UNIVERSIDADE FEDERAL DO CEARÁ
Campus de Quixadá
Prof. Thiago Werlley Bandeira da Silva
QXD0149- Técnicas de Programação Embarcados I

Lista
2025.1

Nome: _____ Matrícula: _____

1. Crie uma macro chamada CLEAR_BIT(x, b) que limpe o bit na posição b da variável x.
2. Crie uma macro chamada `swap_bits(x, i, j)` que troque os bits nas posições `i` e `j` da variável `x`.
3. Considere `unsigned char status = 0b11101111;`. Escreva uma linha de código que zere o bit 4.
4. Escreva uma função `unsigned char mirror_bits(unsigned char value)` que inverta os bits de um byte.
5. Implemente uma função `int compare_bits(unsigned int a, unsigned int b)` que retorna o número de bits iguais entre dois inteiros.
6. Crie uma macro `MASK_BITS(x, m)` que aplique uma máscara `m` sobre `x` e retorne o resultado.
7. Implemente uma função `unsigned int circular_right(unsigned int x, int n)` que rotacione `x` para a direita `n` vezes.
8. Escreva uma **macro** que retorne **TRUE** se seu parâmetro for divisível por 10 e **FALSE** caso contrário.
9. Escreva uma **macro** is_digit que retorne **TRUE** se seu argumento for um dígito decimal.
10. Escreva uma segunda **macro** is_hex que retorne **TRUE** se seu argumento for um dígito hexadecimal (0-9, A-F, a-f). A segunda **macro** deve fazer referência à primeira.
11. Escreva uma **macro** de **pré-processador** que troque(swap) dois inteiros. (Para o hacker real, escreva um que não use uma variável temporária declarada fora da **macro**.)
12. Qual a saída gerada pelo Código 1? Justifique sua resposta.

```c
#include <stdio.h>
#define FIRST_PART  7
#define LAST_PART   5
#define ALL_PARTS FIRST_PART + LAST_PART
int main() {
  printf("The square of all the parts is %d\n", ALL_PARTS * ALL_PARTS);
  return (0);
}
```

Código 1

Marque a resposta correta:

(a). 12
(b). 24
(c). 47
(d). 144

Nota: _____

(e). N.D.A

13. O Código 2 deve imprimir a seguinte mensagem: *"Erro Fatal: Abortar"*e sair quando receber dados incorretos. Mas quando obtém bons dados, ele sai. Por que?

```c
#include <stdio.h>
#include <stdlib.h>
#define DIE \
fprintf(stderr, "Fatal Error:Abort\n");exit(8);
int main() {
  /* a random value for testing */
  int value;
  value = 1;
  if (value < 0)
  DIE;
  printf("We did not die\n");
  return (0);
}
```

Código 2

14. O que o Código 3 produz? Tente executá-lo em sua máquina. Por que produziu o que fez? Tente verificar a saída do pré-processador.

```c
#include <stdio.h>
#define SQR(x) (x * x)
int main(){
  int counter; /* counter for loop */
  for (counter = 0; counter < 5; ++counter) {
    printf("x %d, x squared %d\n",counter+1, SQR(counter+1));
  }
  return (0);
}
```

Código 3

15. Por que o Código 4 não produzirá o resultado esperado? Quanto o contador subirá a cada vez?

```c
#include <stdio.h>
#define SQR(x) ((x) * (x))
int main(){
  int counter; /* counter for loop */
  Counter = 0;
  while (counter < 5)
  printf("x %d square %d\n", counter, SQR(++counter));
  return (0);
}
```

Código 4

16. O Código 5 nos diz que temos uma variável indefinida *number*, mas nosso único nome de variável é *counter*.

```c
#include <stdio.h>
#define RECIPROCAL (number) (1.0 / (number))
int main() {
  float counter; /* Counter for our table */
```

```
5     for (counter = 1.0; counter < 10.0;counter += 1.0) {
6       printf("1/%f = %f\n", counter, RECIPROCAL(counter));
7     }
8     return (0);
9   }
```

Código 5

17. O Código 6 gera um aviso de que o *counter* é usado antes de ser configurado. Este aviso é uma surpresa para nós porque o **for** *loop* deve configurá-lo. Também recebemos um aviso muito estranho, "null effect," para a linha 7.

```
1   #include <stdio.h>
2   #define MAX =10
3
4   int main(){
5     int counter;
6
7     for (counter =MAX; counter > 0; —counter)
8     printf("Hi there\n");
9
10    return (0);
11  }
```

Codigo 6

18. O Código 7 calcula o valor errado para SIZE. Por que?

```
1   #include <stdio.h>
2   #define SIZE 10;
3   #define FUDGE SIZE -2;
4
5   int main(){
6     int size;
7
8     size = FUDGE;
9     printf("Size is %d\n", size);
10    return (0);
11  }
```

Código 7

19. Traduzir os seguintes números hexadecimais para binários.

    0x0, 0x10, 0xF, 0x1F, 0xA4, 0xff

20. Encontrar o bitwise e, ou e xor dos seguintes casos:

    (a) 0xC6 com 0x35
    (b) 0x19 com 0x24
    (c) 0xd3 com 0xC7
    (d) 0x17 com 0xff

21. Encontrar o complemento 1 do seguinte: 0xC6, 0x35, 0xd3 e 0xC7.

22. Nesta pergunta & é bitwise e, | é bitwise ou, $\wedge$ é bitwise xor, e ! é o complemento de 1. a é qualquer dado número hexadecimal de dois dígitos. Explique por que cada uma das seguintes identidades se mantém.

    (a) 0xff & a = a. (0xff é a identidade de E)

(b) 0xff | a = 0xff. (0xff é o absorvente para OR)

(c) 0xff ∧ a = !a

(d) 0 & a = 0. (0 é o absorvente para E)

(e) 0 | a = a. (0 é a identidade da RUP)

(f) 0 ∧ a = a. (0 é a identidade do XOR)

(g) a ∧ a = 0 (a é seu próprio inverso sob XOR)

(h) Para quaisquer três algarismos hexadecimais a, b e c:
Se a ∧ b = c então a ∧ c = b.

23. Escreva um programa que conte o número de bits definido em um número inteiro. Por exemplo, o número 5 (decimal), que é 0000000000000101 (binário), tem dois bits definidos.

24. Escreva um programa que tenha um inteiro de 32 bits (int long) e o divida em oito valores de 4 bits. (Tenha cuidado com o bit de sinal.)

25. Escreva um programa que pegue nos bits de um número e os desloque para a extremidade esquerda. Por exemplo, 01010110 (binário) tornará 11110000 (binário).