Oficina de Programação em C (ICP037)

Prof. Ronald Souza – IC/UFRJ

Primeira Prova – 13/05/2024

GABARITO

Questão 1) (1.0 ponto)

Deseja-se implementar uma função que, para 4 inteiros de entrada, correspondentes a início (*i*), fim (*f*), posição atual (*pa*) e deslocamento (*d*), retorne uma posição nova *pn = pa + d*, ***de forma circular***.

Por exemplo:

ENTRADA				**SAÍDA**
i	*f*	*pa*	*d*	*pn*
90	200	191	7	198
90	200	192	7	199
90	200	193	7	200
90	200	194	7	90
90	200	195	7	91

Assume-se que os valores de entrada são válidos, ou seja, que já estão defendidos externamente. Portanto, *é garantido que i <= f, i <= pa <= f, e d >= 0*. Foi iniciada a escrita da função, conforme o código abaixo:

```
int pos(int i, int f, int pa, int d) {
    // O que retornar?
}
```

→ No seu caderno de respostas **(e não aqui!)**, indique claramente, dentre as alternativas abaixo, a que contém o corpo da função que captura corretamente o comportamento acima descrito. Para isso, **escreva a letra escolhida e também todo o seu conteúdo**:

```
A) return ((pa + d - f) % (f - i + 1) ) + f;
B) return ((pa + d + i) % (f - i + 1) ) - i;
C) return ((pa + d - i) / (f - i + 1) ) + i;
D) return ((pa + d - i) % (f - i + 1) ) + i;
E) return ((pa + d + f) % (f - i + 1) ) - f;
```

RESPOSTA da Questão 1:

LETRA D) `return ((pa + d - i) % (f - i + 1) ) + i;`

Comentários:

`pa + d` → **deslocamento** desejado, que pode ou não vir a sofrer o efeito circular.

`(f - i + 1)` → **total de possibilidades** para qualquer deslocamento circular.

`pa + d - i` → força que a posição inicial seja 0 para facilitar o cálculo do deslocamento circular, que se baseia no **resto da divisão** pelo **total de possibilidades** (o que pode ser zero!). *Por isso mesmo tivemos que readicionar i ao fim!*

Questão 2) (2.5 pontos)

Escreva um **programa** em C que leia do teclado 2 inteiros *a* e *b* **não negativos** e imprima "verdadeiro" caso a soma dos dígitos de *a* seja igual à soma dos dígitos de *b*, ou imprima "falso", caso contrário. Seu programa deve necessariamente conter as duas funções a seguir:

- ***main***: onde os inteiros são lidos do teclado e o resultado final é impresso.
- ***somaDig***: função que recebe um único inteiro de entrada e retorna a soma de seus dígitos.

Por exemplo, as entradas abaixo produzirão as seguintes saídas:

ENTRADA		**SAÍDA**
a	***b***	
1021	31	"verdadeiro" (pois 1+0+2+1 = 3+1 = 4)
684	9009	"verdadeiro" (pois 6+8+4 = 9+0+0+9 = 18)
58	44	"falso" (pois 5+8 = 13, mas 4+4 = 8)

RESPOSTA da Questão 2:

```
/* Questão 2
   Entrada: 2 inteiros a e b não negativos.
   Saída: Mensagem "verdadeiro" se a soma dos dígitos for igual; "falso" c.c.
   Defesa: a >= 0; b >= 0.
*/
int somaDig(int x); //Protótipo
int main() {
    int a, b;
    printf("Inteiros a e b, separados por espaco: ");
    scanf("%d %d", &a, &b);

    //Defesa
    if (a < 0 || b < 0) {
       printf("Valores invalidos");
       return -1;
    }
    puts((somaDig(a) == somaDig(b)) ? "verdadeiro" : "falso"); //Chamada
    return 0;
}

int somaDig(int x) { //Implementação
    int soma = 0;
    while (x) {
       soma += x % 10;
       x /= 10;
    }
    return soma;
}
```

Questão 3) (1.5 pontos)
Escreva uma **função** que receba dois inteiros *x* e *y* de entrada. Assuma que *x* e *y* **já estão defendidos**, de tal forma que, garantidamente, *x >= 0, y >= 0* e *x < y*. Sua função deverá **imprimir em ordem crescente** todos os inteiros que são múltiplos, **ao mesmo tempo**, de 2 e de 3, dentro do intervalo fechado [*x*, *y*] (i.e. incluindo-se *x* e *y*, **se for o caso**). Sua função **não deve retornar valores**.

RESPOSTA da Questão 3:
Como ser múltiplo de 2 e 3 ao mesmo tempo implica ser múltiplo de 6, podemos fazer como segue:

```
//Opção 1 - ajuste inicial de x, incremento de 6 em 6:
void multiplos_2e3(int x, int y) {
    while (x % 6) x++;
    for (int i = x; i <= y; i += 6)
       printf("%d ", i);
}
```

```
//Opção 2 - força bruta - testar cada elemento; incremento unitário:
void multiplos_2e3(int x, int y) {
    for (int i = x; i <= y; i++)
       if(i % 6 == 0)
          printf("%d ", i);
}
```

```
//Opção 3 (curiosidade) - aplicar fórmula; incrementar de 6 em 6 (mais eficiente):
void multiplos_2e3(int x, int y) {
    for (int i = 6 * ceil(x/6.0); i <= y; i += 6)
       printf("%d ", i);
}
```

COMENTÁRIOS SOBRE A OPÇÃO 3:

- ceil() é a função da biblioteca math.h do C que computa o **teto** de um argumento de entrada
- De um modo geral, um número *x* é simultaneamente múltiplo de um conjunto A qualquer de inteiros primos entre si se *x* é múltiplo do produtório dos valores de A. Assim, se quisermos imprimir todos os números que são múltiplos de *z1, z2, z3, … , zn*, dentro de um intervalo [*x, y*] onde *x <= y* e *z1, z2, z3, … , zn* são primos entre si, basta encontrarmos o valor *k = z1* z2 * … * zn* e daí
    - obtermos o 1o elemento da série, que será *k* * teto(*x/k*);
    - encontrarmos os demais elementos, com passos de tamanho *k.*

- Note que, na questão acima, *k* = 3*2 = 6.
- Note que x é inteiro. Assim, dentro da função ceil() não dividimos x por 6 (literal inteiro) mas por 6.0 (literal racional), para que o C compute uma divisão racional.

Questão 4) (2.5 pontos)

Embora o ouvido humano perceba frequências sonoras que vão desde 20 Hertz (Hz) até 20000 Hz, sua sensibilidade é maior para frequências intermediárias, entre 2000 e 5000 Hz.

Ou seja, para um mesmo volume de reprodução, sons entre 2000 e 5000 Hz são os mais intensamente *audíveis* por seres humanos, em comparação às demais frequências do espectro auditivo. Assumindo-se valores ***já defendidos***, execute as seguintes tarefas:

a) (1.0 ponto) Escreva uma **função** que receba 3 racionais de entrada, correspondentes a frequências sonoras quaisquer *f1*, *f2*, e *f3*, e retorne o inteiro

   1, caso **ao menos uma** dentre as frequências de entrada pertença à faixa mais audível;

   0, caso contrário.

b) (1.5 pontos) Assuma que a frequência sonora mais sensível ao ouvido humano seja 3500 Hz. Escreva uma **função** que receba 3 racionais de entrada, correspondentes a frequências sonoras quaisquer *f1, f2,* e *f3*, e retorne a mais audível dentre elas, ou seja, a frequência que está **mais próxima de 3500Hz**.

RESPOSTA da Questão 4:

```
//Questão 4.a)
int contido(double f1, double f2, double f3) {
    const double I = 2000, S = 5000; //Limites inferior (I) e superior (S)
    return f1 >= I && f1 <= S || f2 >= I && f2 <= S || f3 >= I && f3 <= S;
}



//Questão 4.b)
double maisProx(double f1, double f2, double f3) {
    const double F = 3500; //Nossa referência
    double d1 = fabs(f1 - F), d2 = fabs(f2 - F), d3 = fabs(f3 - F);
    return (d1 < d2 && d1 < d3) ? f1 : (d2 < d3) ? f2 : f3;
}
```

Questão 5) (2.5 pontos) O que será impresso ao final de cada iteração do programa abaixo? No seu caderno de respostas, faça uma tabela análoga a descrita a seguir, e a preencha com os valores esperados:

```
#include <stdio.h>
int main() {
	float a = 1, b = 1;
	int x = 1;
	for (int i = 1; i <= 3; i++) {
		b += i + i / 2;
		a = i + b / 2;
		x = b - a;
		if (x && i) {
			printf("a = %.1f; b = %.1f; x = %d\n", a, b, x);
			continue;
			++b;
		}
		a = x / 2;
		printf("a = %.1f; b = %.1f; x = %d\n", a, b, x);
	}
	return 0;
}
```

i	a	b	x
1	?	?	?
2	?	?	?
3	?	?	?

RESPOSTA da Questão 5:

i	**a**	**b**	**x**
1	0.0	2.0	0
2	0.0	5.0	0
3	7.5	9.0	1