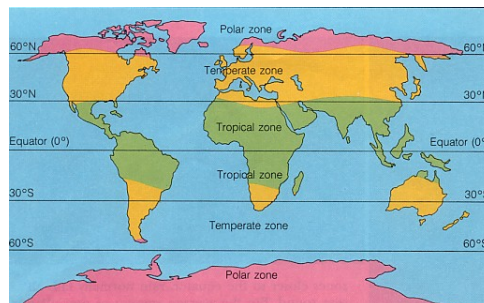


**Lista 3 – Computação Científica I (INE5231) – 2017s1**  
**Engenharia Mecânica – Universidade Federal de Santa Catarina**

- Escreva um programa, em C, para manipulação de datas, com três inteiros: dia, mês e ano. Com base nas entradas da data de nascimento de uma pessoa e da data atual, pede-se:
  - Calcule a idade exata da pessoa
  - Verifique se a pessoa é maior de idade
- A partir da leitura de uma latitude, escreva um programa, em C, que informe a zona térmica correspondente. Seguem os intervalos aproximados:
  - Latitude entre  $-30^\circ$  a  $30^\circ$  para “tropical”
  - Latitude entre  $-60^\circ$  a  $-30^\circ$  e entre  $30^\circ$  a  $60^\circ$  para “temperado”
  - Latitude entre  $-90^\circ$  a  $-60^\circ$  e entre  $60^\circ$  a  $90^\circ$  para “polar”



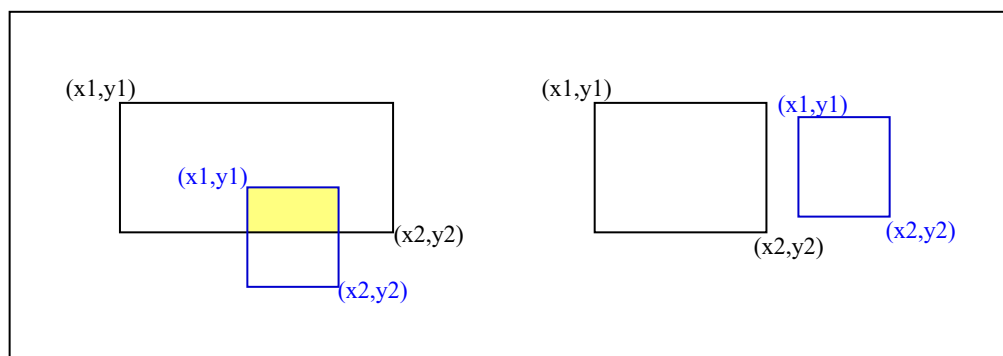
<http://www.webquest.hawaii.edu/kahihi/sciencedictionary/images/degreeemap.jpg>

- A temperatura média ( $^\circ\text{C}$ ) de Florianópolis é a seguinte (Wikipedia):

Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez	Ano
24,2	24,6	23,6	21,1	18,9	16,7	16,5	16,9	17,9	19,6	21,3	23	20,4

Escreva um programa, em C, que, dado o número do mês (1 a 12), devolva a temperatura média da cidade. Se o número do mês for diferente de 1 a 12, devolva a média anual. Utilize preferencialmente o switch-case (veremos vetores mais adiante).

- Considere um retângulo, alinhado com os eixos do plano cartesiano, representado por duas coordenadas:  $(x_1, y_1)$  do canto esquerdo superior, e  $(x_2, y_2)$  do canto direito inferior (na orientação do eixos,  $x_1 < x_2$  e  $y_1 > y_2$ ). Escreva um programa, em C, que leia essas duas coordenadas de dois retângulos (8 valores reais), e exiba, na tela, as coordenadas do retângulo da interseção (4 valores reais). Se não houver interseção, exiba a mensagem “intersecao inexistente”.



---

**Dicas:** os pontos do novo retângulo podem ser calculados com regras simples, implementadas por meio de `if`'s encadeados e operações de mínimo e máximo. Nem sempre existe intersecção entre dois retângulos. Considere a figura acima. No exemplo do lado esquerdo existem dois retângulos (um com contorno em preto, o outro em azul) que têm intersecção dada pela área preenchida (em amarelo), e, no lado direito, dois que não têm. O programa deve determinar o retângulo da intersecção (se houver).

5. Imprima todos os números de 150 a 300.

6. Imprima a soma de 1 até 1000.

7. Imprima todos os múltiplos de 3, entre 1 e 100.

8. Imprima os fatoriais de 1 a 10.

O fatorial de um número  $n$  é  $n * n-1 * n-2 \dots$  até  $n = 1$ . Lembre-se de utilizar os parênteses.

O fatorial de 0 é 1

O fatorial de 1 é  $(0!) * 1 = 1$

O fatorial de 2 é  $(1!) * 2 = 2$

O fatorial de 3 é  $(2!) * 3 = 6$

O fatorial de 4 é  $(3!) * 4 = 24$

Faça um `for` que inicie uma variável  $n$  (número) como 1 e fatorial (resultado) como 1 e varia  $n$  de 1 até 10:

```
1 | int fatorial = 1;
2 | for (int n = 1; n <= 10; n++) {
3 |
4 | }
```

9. No código do exercício anterior, aumente a quantidade de números que terão os fatoriais impressos, até 20, 30, 40. Em um determinado momento, além desse cálculo demorar, vai começar a mostrar respostas completamente erradas. Por quê?

Mude de `int` para `long` para ver alguma mudança.

10. Imprima os primeiros números da série de Fibonacci até passar de 100. A série de Fibonacci é a seguinte: 0, 1, 1, 2, 3, 5, 8, 13, 21, etc... Para calculá-la, o primeiro elemento vale 0, o segundo vale 1, daí por diante, o  $n$ -ésimo elemento vale o  $(n-1)$ -ésimo elemento somado ao  $(n-2)$ -ésimo elemento (ex:  $8 = 5 + 3$ ).

11. Escreva um programa que, dada uma variável  $x$  com algum valor inteiro, temos um novo  $x$  de acordo com a seguinte regra:

- se  $x$  é par,  $x = x / 2$
- se  $x$  é ímpar,  $x = 3 * x + 1$

- imprime  $x$
- O programa deve parar quando  $x$  tiver o valor final de 1. Por exemplo, para  $x = 13$ , a saída será:  
40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1

12. Imprima a seguinte tabela, usando for's encadeados:

```

1
2 4
3 6 9
4 8 12 16
n n*2 n*3 . . . . n*n

```

13. Leia um número  $n$  e escreva o quadrado e o cubo para inteiro de 2 a  $n$ .

14. Leia  $n$  números inteiros e, em seguida, escreva quantos são pares e quantos são ímpares.

15. Leia  $n$  números inteiros e, em seguida, escreva qual o maior deles.

16. Para os números inteiros de 1000 a 9999, imprimir os que têm a característica do exemplo a seguir:

- $30 + 25 = 55$
- $55^2 = 3025$

17. Leia um número inteiro e some todos os seus divisores.

18. Suponha que você invista seu dinheiro a juros fixos de  $r\%$  ao mês. Após  $n$  meses, o seu investimento crescerá segundo a seguinte fórmula:

Número de meses	Investimento acumulado
1	$a + (r \times a) = a(1 + r)$
2	$a(1 + r) \times (1 + r) = a(1 + r)^2$
3	$a(1 + r)^2 \times (1 + r) = a(1 + r)^3$
$\vdots$	
$n$	$a(1 + r)^{n-1} \times (1 + r) = a(1 + r)^n$

Calcule e escreva a tabela acima, dado um investimento inicial  $a$ , um número  $n$  de meses e juros de  $r\%$ .

19. Crie uma classe para guardar um número inteiro positivo  $n$  com os seguintes métodos:

- $inv(n)$ : inverte a ordem dos dígitos de  $n$ . Exemplo:  $inv(782) = 287$ .
- $palindromo(n)$ : verifica se  $n$  é ou não palíndromo, ou seja, se  $inv(n) == n$ . Exemplos de palíndromos: 34543, 1, 99.
- $primo(n)$ : verifica se  $n$  é primo, ou seja, que tem apenas dois divisores diferentes (1 e ele mesmo). Exemplos de primos: 2, 3, 5, ..., 577, ..., 997.

---

20. Escreva um programa, em Java, para o cálculo do máximo divisor comum (*MDC*) entre dois números inteiros positivos. O algoritmo de Euclides para o *MDC* entre  $M$  e  $N$  é dado por:

- Se  $N > M$ , retorne  $MDC(N, M)$
- Se  $N == 0$ , retorne  $M$
- Senão, retorne  $MDC(N, M \% N)$  (onde  $\%$  é o operador de resto da divisão inteira)

21. Escreva um programa, em Java, para o cálculo da função de Ackermann –  $A(x, y)$  –, dada por:

- Se  $x == 0$ ,  $A(x, y) = y + 1$
- Se  $y == 0$ ,  $A(x, y) = A(x - 1, 1)$
- Senão,  $A(x, y) = A(x - 1, A(x, y - 1))$

Evite calcular o valor desta função para valores grandes de  $x$  e/ou  $y$ . Por exemplo,  $A(3, 4)$  gera mais de 10000 chamadas recursivas. Veja outras curiosidades em:  
[http://pt.wikipedia.org/wiki/Função\\_de\\_Ackermann](http://pt.wikipedia.org/wiki/Função_de_Ackermann)