6.1 Utilizando a função `rand()` da biblioteca padrão de C, escreva pequenos programas que simulem uma escolha aleatória os seguintes valores:

(a) um inteiro entre 0 e 9;

(b) um inteiro entre 1 e 10;

(c) um inteiro entre 1 e 100;

(d) um inteiro entre -10 e 10;

(e) um `double` entre 0.0 e 1.0;

(f) um `double` entre -1.0 e 1.0;

(g) um inteiro par entre 0 e 100;

(h) um inteiro ímpar entre 1 e 99.

Sugestão: veja os exemplos para gerar valores pseudo-aleatórios num intervalo apresentados na aula 11. No caso de valores inteiros, os dois extremos do intervalo devem estar incluídos; para valores vírgula-flutuante, o extremo inferior desde estar incluído e o superior não.

Para testar as suas soluções faça um ciclo que as execute e imprima várias vezes (por exemplo, 100 vezes).

6.2 Escreva um programa que põe à prova os conhecimentos sobre a tabuada e gerando 10 perguntas aleatórias do tipo "`Quanto é 3 x 9?`". O utilizador introduz um valor e o programa diz se está "`Certo!`" ou se está errado. Em caso de erro o programa responde com uma mensagem "`Errado! O resultado é 27.`". Os inteiros nas perguntas devem estar compreendidos entre 1 e 9 inclusivé. No final o programa deve indicar o número de respostas certas e erradas. O número de perguntas (10) deve ser fácil de modificar por meio de `#define`.

6.3 Escreva um programa para jogar *Advinha o Número*: o computador escolhe aleatoriamente um número inteiro secreto entre 1 e 1000. Em seguida pede ao jogador humano que introduza uma tentativa; se a tentativa for menor que número secreto escreve "`Demasiado baixo!`"; se for maior, escrever "`Demasiado alto!`". Enquanto o jogador não acertar no número secreto vai pedido novas tentativas. Quando o jogador acertar, escreve "`Acertou em ... tentativas!`" e termina o programa.

(Depois de resolver o exercício pense nas seguintes questões: Qual é a melhor estratégia para escolher a próxima tentativa? E qual é o número mínimo de tentativas tal que garantidamente consegue acertar no número secreto?)

▷ **6.4** Escreva uma definição da função

```
void range(int vec[], unsigned size, int inicio, int incr)
```

que inicializa elementos de um vector `vec` com `size` valores inteiros `inicio`, `incio+incr`, `incio+2*incr`, etc. seguindo uma progressão aritmética. Exemplo:

```
int a[5];
range(a, 5, 3, 2); /* a[] passa a conter { 3, 5, 7, 9, 11 } */
```

Utilize um ciclo `for` para percorrer os índices válidos e atribuir valores sucessivos aos elementos do vector.

6.5 Escreva uma função `int ocorre(int vec[], int size, int val)` que procura um valor `val` no elementos de um vector `vec`. Se o valor ocorre como um dos elementos do vector, o resultado deve ser 1; caso contrário deve ser 0.

6.6 Escreva um programa que lê uma sequência de valores inteiros positivos terminada por -1 e no final escreve toda a sequência mas sem repetidos.

Sugestão: guarde os valores numa variável indexada à medida que são lidos; de cada vez que lê um novo valor pode verificar se já é repetido (e nesse caso não o guarde novamente). No final imprima todos os valores guardados. Pode usar a função do exercício 6.5 para testar se um valor lido ocorre entre os anteriores (i.e., se é repetido).

6.7 Escreva um programa que lê carateres e contabiliza o número de vezes de ocorreu cada letra (A a Z) ignorando a distinção entre maiúsculas e minúsculas. No final deve imprimir uma tabela com as contagens:

```
A: 3    B: 2    C: 2    D: 1    E: 7    ...
```

Sugestão: utilize uma variável indexada com 26 elementos para contabilizar a contagem das letras. Pode ainda converter cada carater para maíuscula para evitar ter de tratar letras minúsculas (ver o último exemplo da aula 10).

▷ **6.8** Escreva uma função `int repetidos(int vec[], unsigned size)` que testa se há (pelo menos) dois valores iguais no vector `vec` com tamanho `size`; o resultado deve ser 1 em caso afirmativo e 0 em caso negativo. Exemplos:

```
int a[5] = { 2, -1, 0, 2, -1 };
int b[5] = { 3, 4, 1, 2, -1 };
printf("%d\n", repetidos(a, 5)); // imprime 1
printf("%d\n", repetidos(b, 5)); // imprime 0
```

Tenha atenção que a sua função não modifique os elementos do vector passado como argumento.

6.9 Numa turma com 25 alunos, qual a probabilidade de que o aniversário de pelo menos dois deles seja no mesmo dia? E se forem 50 alunos? O *paradoxo dos aniversários* é a constantação de que esta probabilidade é maior do que parece ao senso comum.[1] Vamos estimar experimentalmente a probabilidade de aniversários repetidos com N alunos escrevendo um programa que repetidamente gere dias aleatoriamente (N valores de 1 a 365) e verifique a ocorrência de repetições.

Utilize `rand()` para gerar aleatoriamente um vector com N dias e a função do exercício 6.8 para testar repetições. Se fizer um grande número de experiências, a frequência relativa de ocorrência de repetidos dá um valor aproximado da probabilidade. O número de alunos N e de experiências deve ser facilmente modificável (por exemplo, usado `#define`).

[1] `http://en.wikipedia.org/wiki/Birthday_paradox`.