Com o VB.NET desenvolver Windows Applications para os problemas a seguir propostos.
Crie as rotinas e uma interface adequada.

1. Um número aleatório é um conceito matemático preciso: cada valor deve ocorrer com igual probabilidade num domínio finito. Este conceito contrasta com o significado vulgar de que um número aleatório é um número arbitrário, que significa um valor qualquer.
Para se obter um aleatório, parte-se do anterior - raiz, multiplica-se por uma constante *B* e soma-se 1 e toma-se o resto da divisão inteira por outra constante *M*. O resultado será sempre um inteiro entre 0 e *M-1*.
Gerar *N* aleatórios.
(Deve dar-se especial atenção à operação de multiplicar uma vez que pode originar um *overflow)*

2. Com base na aplicação anterior desenvolva outra que gere 10000 aleatórios entre 1 e 1000, que registe as ocorrências por grupos de 20 e que faça um **gráfico** de barras horizontais de caracteres com as ocorrências por cada grupo.

3. Suponha que num sistema operativo as tarefas aparecem a todo o momento e o processador vai pedindo novas tarefas para executar. Em cada momento a tarefa que deve ser posta em funcionamento é a de maior prioridade. Pode criar-se uma estrutura – *heap*, para suportar o problema. Num *heap* os elementos estão arranjados em sequência, de tal modo que cada um deve ser maior ou igual do que os dois seguintes ainda não considerados. Com esta organização está sempre disponível o elemento de maior prioridade sem termos o custo temporal de uma ordenação, cada vez que aparece uma nova tarefa.
A aplicação vai receber N valores de prioridades, coloca-os num ***heap***, e que retira-os em sequência.

4. Escrever uma função que faça o cálculo da potência de um número de forma recursiva.
Recordemos que: **X^N ⇔ X.X^(N-1)**

5. Escrever um procedimento recursivo que implemente o método de ordenação designado por Quicksort.
O QuickSort (1960, C.A.R. Hoare) é um algoritmo que usa a técnica "dividir para reinar". Funciona partindo a colecção de valores em duas partes e ordenando-as separadamente. O local da partição depende da colecção. A ideia base é a seguinte:

```
ORDENA (ESQUERDA, DIREITA) é:
  PONTOPARTIÇÃO = PARTIÇÃOENTRE(ESQUERDA, DIREITA);
  ORDENA (ESQUERDA, PONTOPARTIÇÃO-1);
  ORDENA (PONTOPARTIÇÃO+1, DIREITA)
```

Os parâmetros ESQUERDA e DIREITA (supõe-se ESQUERDA < DIREITA) delimitam a subcolecção a ordenar dentro da colecção original. Da primeira vez o algoritmo é chamado com ORDENA(1,N).

O ponto crucial é a PARTIÇÃOENTRE os limites à ESQUERDA e à DIREITA e que deve observar as seguintes condições:

a) O elemento A[PONTOPARTIÇÃO] está na sua posição final na colecção, e será ignorado das vezes seguintes;
b) Todos os elementos desde A[ESQUERDA], ..., A[PONTOPARTIÇÃO-1] são menores ou iguais a A[PONTOPARTIÇÃO];
c) Todos os elementos desde A[PONTOPARTIÇÃO-1], ...,A[DIREITA] são maiores ou iguais a A[PONTOPARTIÇÃO];

Isto pode ser implementado de um modo muito simples: Primeiro escolhe-se um A[DIREITA] arbitrário, elemento que irá para a sua posição final. Depois varre-se a partir da esquerda até se encontrar um elemento maior do que A[DIREITA] e varre-se a partir da direita até se encontrar um elemento menor do que A[DIREITA]. Estes dois elementos que provocaram a paragem estão obviamente fora do seu lugar e devem ser trocados. (Deve também parar-se o varrimento se forem detectados elementos iguais ao do PONTOPARTIÇÃO). Continuando desta forma asseguramos que todos os elementos à esquerda do apontador esquerdo são menores que A[DIREITA], e todos os elementos à direita são maiores. Quando os apontadores de varrimento se cruzam troca-se A[DIREITA] com o elemento à esquerda da subcolecção da direita.

6. Implemente um procedimento para o cálculo recursivo da raiz de um número.
Matematicamente o problema coloca-se do seguinte modo: dado um X > 0, encontrar Y tal que Y^2 = X.
Sabemos o que procuramos, mas ainda não temos o algoritmo. Podemos pensar que se Y^2 = X, então X / Y = Y, o que nos dá uma ideia para o algoritmo:
- Estimar um valor R para Y e testá-lo
- Calcular X / R
- Se X / R é suficientemente próximo de R, então retorna-se R. Caso contrário deve estimar-se melhor.

A questão de saber se se está suficientemente próximo prende-se com o cálculo a menos de um erro, que pode ser o valor que se quiser desde que pequeno (ex: 0,0001)
Estimar melhor pode ser o principal problema. Deve encontrar-se um valor mais próximo da raiz de X do que R. Uma ideia será se X / R não estiver próximo de R estimar o novo valor como a média entre X / R e R.

7. Um número aleatório é um conceito matemático preciso: cada valor deve ocorrer com igual probabilidade num domínio finito. Este conceito contrasta com o significado vulgar de que um número aleatório é um número arbitrário, que significa um valor qualquer.
Para se obter um aleatório, parte-se do anterior - raiz, multiplica-se por uma constante *B* e soma-se 1 e toma-se o resto da divisão inteira por outra constante *M*. O resultado será sempre um inteiro entre 0 e *M-1*.
Gerar *N* aleatórios.
(Deve dar-se especial atenção à operação de multiplicar uma vez que pode originar um *overflow)*

8. Com base na aplicação anterior desenvolva outra que gere 10000 aleatórios entre 1 e 1000, que registe as ocorrências por grupos de 20 e que faça um **gráfico** de barras horizontais de caracteres com as ocorrências por cada grupo.

9. Suponha que num sistema operativo as tarefas aparecem a todo o momento e o processador vai pedindo novas tarefas para executar. Em cada momento a tarefa que deve ser posta em funcionamento é a de maior prioridade. Pode criar-se uma estrutura – *heap*, para suportar o problema. Num *heap* os elementos estão arranjados em sequência, de tal modo que cada um deve ser maior ou igual do que os dois seguintes ainda não considerados. Com esta organização está sempre disponível o elemento de maior prioridade sem termos o custo temporal de uma ordenação, cada vez que aparece uma nova tarefa.
A aplicação vai receber N valores de prioridades, coloca-os num ***heap***, e que retira-os em sequência.

10. Crie uma estrutura para um aluno, leia para um vector alguns elementos, ordene o vector por um dos campos e faça uma pesquisa binária para um aluno pedido, mostrando todos os seus atributos.

11. Desenvolva uma rotina capaz de converter um nome completo para o formato último_nome, primeiro_nome iniciais_dos_restantes_nomes

Exemplo:
Nome completo: António José Martins Costa
Nome convertido: Costa, António J. M.

12. Desenvolva um programa que leia *strings* com no máximo 100 letras (entre 'A' e 'Z') e ordene as letras de acordo com a ordem em que aparecem na *string* ignorando os espaços.
Por exemplo se a *string* for "TECNICAS DE PROGRAMACAO" o resultado será "TEEECCCNIAAAASDPRROOGM".

13. Desenvolva um programa que codifique um texto utilizando o seguinte método: cada letra de uma dada palavra é substituída pela correspondente distanciada de N posições no alfabeto, sendo N o comprimento da palavra em causa. Considere que a letra a seguir à última é a primeira.

Exemplo: TECNICAS DE PROGRAMACAO
BMKVQKIA FG ACZRCLXLNLZ

Dado que a palavra TECNICAS tem comprimento 8, a letra correspondente a T é B (distanciada de 8 posições no alfabeto), a letra correspondente a E é M, e assim sucessivamente.

14. Desenvolva um programa que leia um texto e permita procurar e substituir palavras no texto. O programa deve permitir as seguintes funcionalidades:

Procurar – procura a ocorrência uma determinada palavra (ou texto) no texto.
Procurar Seguinte - procura a próxima ocorrência da palavra (ou texto) no texto.
Substituir – procura e substitui uma palavra (ou texto) por outra.

15. Desenvolva uma rotina capaz de: Ler um ficheiro de texto e contar as ocorrências de cada carácter apresentando o resultado numa lista ou num outro ficheiro (deve pedir o nome).
Nota: Deve procurar trabalhar com um ficheiro de texto o maior possível e apenas com palavras em Português.

16. Desenvolva uma rotina capaz de: Ler um ficheiro de texto e para cada carácter contar as ocorrências do carácter seguinte, i.e., o 'a' é seguido 10 vezes pelo 'a', 254 vezes pelo 'b', 220 vezes pelo 'c', ...; o 'b' é seguido 437 vezes pelo 'a', 0 vezes pelo 'b', ...
Apresente o resultado numa grelha e num ficheiro de texto.

17. Com base no anterior, crie uma cadeia de 100 caracteres para cada carácter, num vector, onde cada carácter que o segue apareça na percentagem das ocorrências contadas. Por exemplo, para o 'b' pode ser:
"aaaaaaaaaaaaaaaaaaaaaaaaaaeeeeeeeeeeeeeeeeeeeeeeeeeiiiiiiiiiiiiijjjllllooooooooorrrrrrrrrrrrrrrrrrrsssss""
se é seguido pelo 'a' em 27% dos casos, pelo 's' em 5% dos casos.
O resultado deve ser mostrado numa lista ou em ficheiro de texto.

18. Com base no vector anterior crie um rotina de geração de palavras aleatórias de um comprimento passado como parâmetro: PALEATO(NCAR: integer): string. Deve gerar um 1º carácter aleatoriamente e depois, até ao comprimento pedido, gerar os seguintes, usando as strings previamente criadas com as probabilidades de ocorrência.