Dayman Moreira Novaes **29/05/2015**
Ciência da Computação 1º Período

Matemática Discreta Trabalho Prático

ANÁLISE COMBINATÓRIA

Introdução
A análise combinatória é um problema matemático relacionado à contagem, mais especificamente em encontrar o que deve ser contado. Uma vez identificados os elementos, conta-los se torna fácil.
Este programa realiza a contagem de todos os elementos, dada duas condições determinadas pelo usuário:

- **Ordem**
  Define se a ordem dos elementos importa. No caso que a ordem não é importante, temos que, por exemplo, a contagem 123 é equivalente a 321, portanto ambos devem ser considerados um só. Caso a ordem seja importante, os dois exemplos anteriores são contados separadamente.
- **Repetição**
  Define se podemos ter elementos repetidos ou não. No caso em que a repetição não é permitida, não podemos contar o elemento 122, pois o "2" repete.

Uma vez determinada as condições, o usuário determina também:

- **Quantidade de elementos**
- **Numero máximo de cada elemento**
  A contagem de cada elemento é feito de 1 ao número escolhido.

Para cada um dos quatro casos (considerando todas as possibilidade para ordem e repetição), temos as seguintes fórmulas:

Repetição	**Ordem**		
		Sim	**Não**
	Sim	n^r	$\binom{n+r-1}{r}$
	Não	$\frac{n!}{(n-r)!}$	$\binom{n}{r}$

Implementação
O programa segue um fluxo simples e bem segmentado, onde cada função (com um nome autoexplicativo), realiza um passo preparando os dados para a próxima ação, atingindo o objetivo no final.

- **Lê dados**
  Pega os dados do usuário

- **Valida dados**
  Verifica se os dados digitados estão nas especificações
  - S ou N para Ordem e Repetição
  - Máximo de 10 itens
  - Número máximo maior ou igual a um e, dependendo do caso, menor que a quantidade de itens.

- **Gera tipo de combinação**
  Dependendo das entradas de Ordem e Repetição, gera um número que representa qual é o caso a ser considerado, facilitando todas as verificações no resto do programa.

- **Impressão de resultados**
  Formada de vários passos, nos quais:
  - **Impressão da quantidade de combinações**
    Baseada nas fórmulas descritas acima, utilizando uma função de fatorial, como mostra a imagem ao lado.

```
double fat(int n) {
    double soma = 1;
    for(n; n>0; n--)
        soma *= n;
    return soma;
}
double or(int n, int r) {
    return pow(n, r);
}
double onr(int n, int r) {
    return fat(n)/fat(n-r);
}
double nor(int n, int r) {
    return fat(n+r-1)/(fat(r)*fat(n-1));
}
double nonr(int n, int r) {
    return fat(n)/(fat(r)*fat(n-r));
}
```

  - **Impressão de todas as combinações possíveis:**
    Esse passo é composto de três simples passos: geração de um novo número de forma incremental, validação se o número gerado se encaixa nas especificações, e impressão (caso encaixado).

```
for(cont; cont>0; cont--) {
    if(numValido(num, r, tipoCombinacao))
        imprimeNum(num, r);

    incrementa(num, n, r);
}
```

A validação verifica o tipo de combinação (gerada no terceiro passo) e o requisito necessário para a combinação, como mostra a imagem.

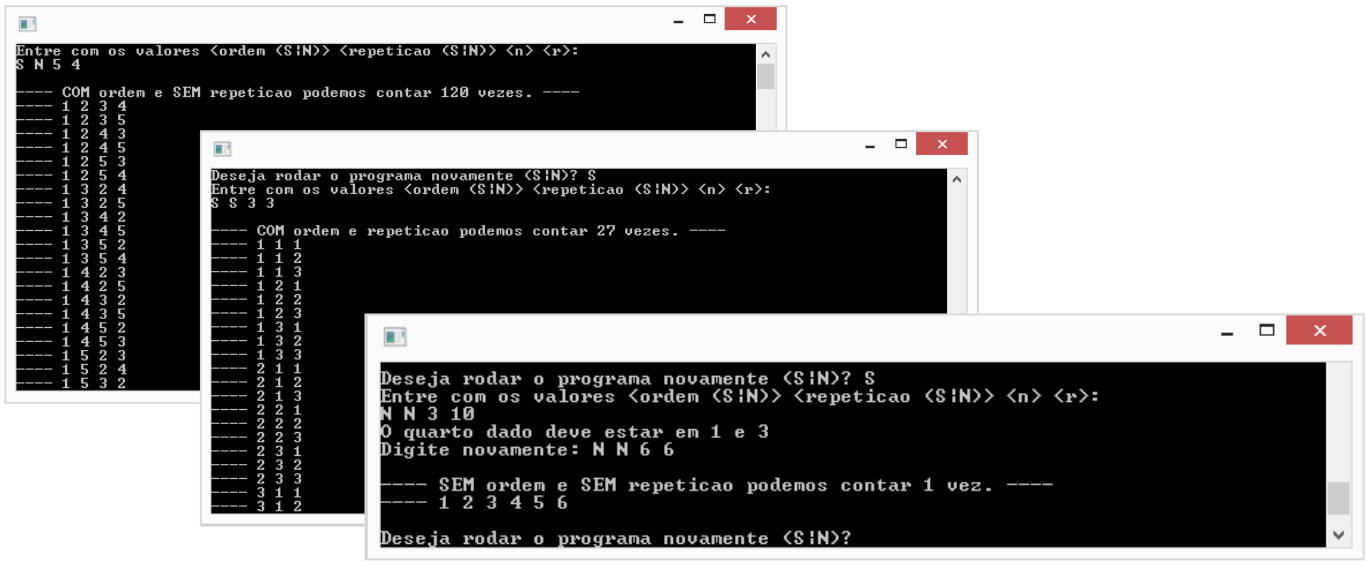
```c
int numValido(int *num, int r, int tipoCombinacao) {
    int c = tipoCombinacao;

    return  c == ordem_repete
        || (c == ordem_nao_repete && !repete(num, r))
        || (c == nao_ordem_repete && !ordem(num, r))
        || (c == nao_ordem_nao_repete && !repete(num, r) && !ordem(num, r));
}
```

A validação se os algarismos repetem é bem intuitiva.

Já a validação se o número atende os requisitos de não poder repetir considerando a ordem é um pouco mais complexo, mas ele se baseia no seguinte princípio:
O número gerado, para ser válido, deve ser o menor número possível a ser formado com os mesmos algarismos. Porque caso exista um número menor, significa que esse número já foi gerado antes (pois a geração é sequencial).

Funcionamento



Conclusão

O programa é executado de forma rápida, até para valores grandes, como conjuntos de 10 números onde cada um varia de 1 a 10, mostrando que a implementação está satisfatória.

A fase de implementação ocorreu sem problemas, porém a maior dificuldade foi achar uma solução se o número gerado se encaixava na especificação “ordem importa”.