Definição

Sequência finita de passos que se corretamente seguidos, nos levam a resultados previsíveis.

Exemplo: Receita para preparar um bolo.

ALGORITMO para preparar um bolo.

1 Início
2 Bata as claras em neve.
3 Misture as gemas, a margarina e o açúcar.
4 Acrescente o leite e a farinha de trigo.
5 Adicione as claras em neve e o fermento.
6 Despeje a massa em uma forma.
7 Asse o bolo um em forno.
8 Repita.
9 Fim

Observações do Algoritmo apresentado acima

1 É a descrição de um procedimento rotineiro;
2 Tem um INÍCIO e um FIM claros;
3 A descrição é feita passo a passo, de maneira bem definida;
4 Há imperfeições:
4.1 Não especifica a quantidade de cada ingrediente;
4.2 Não especifica quantas tempo o processo 7 deve durar;
4.3 Não especifica qual o processo ou qual passo que deve ser repetido e nem em quis circunstâncias eles devem ser repetidos.

Com isso, verificamos que enquanto houverem imperfeições e dúvidas, o Algoritmo deve ser melhorado.

Melhorando o Algoritmo:

ALGORITMO para preparar um bolo.

1 Início
2 Bata três claras em neve.
3 Misture as três gemas, 48g de margarina e 320g açúcar.

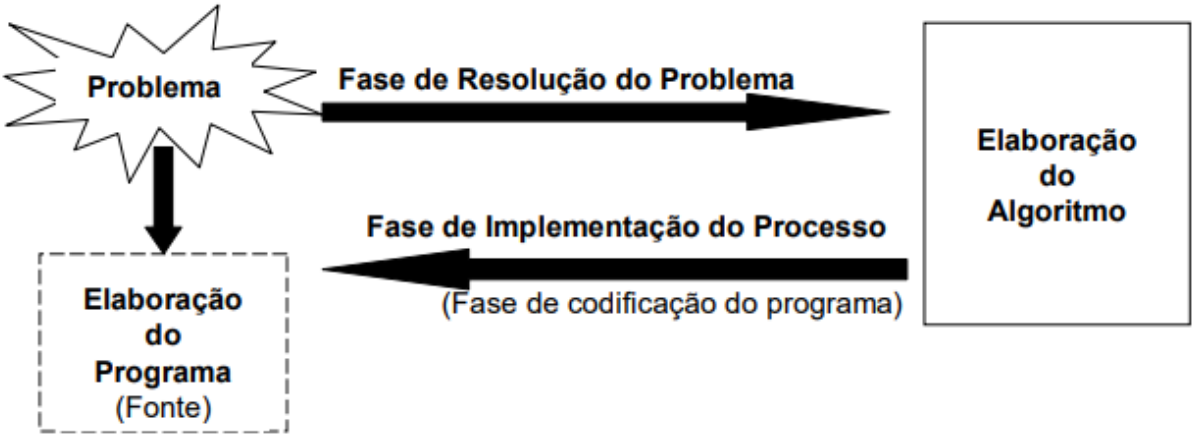
4 Acrescente 360ml de leite e 360g de farinha de trigo.
5 Adicione as claras em neve e 30g de fermento.
6 Despeje a massa em uma forma.
7 Asse o bolo um em forno durante 40 minutos.
8 Repita os processos 2-7 caso queira mais um bolo.
9 Fim

Características

Para que servem os Algoritmos em Computação?

Servem para a elaboração do programa fonte. Está antes do fluxograma acima.

Serve para sairmos do problema e chegarmos ao programa.



Qualidades de um bom Algoritmo:

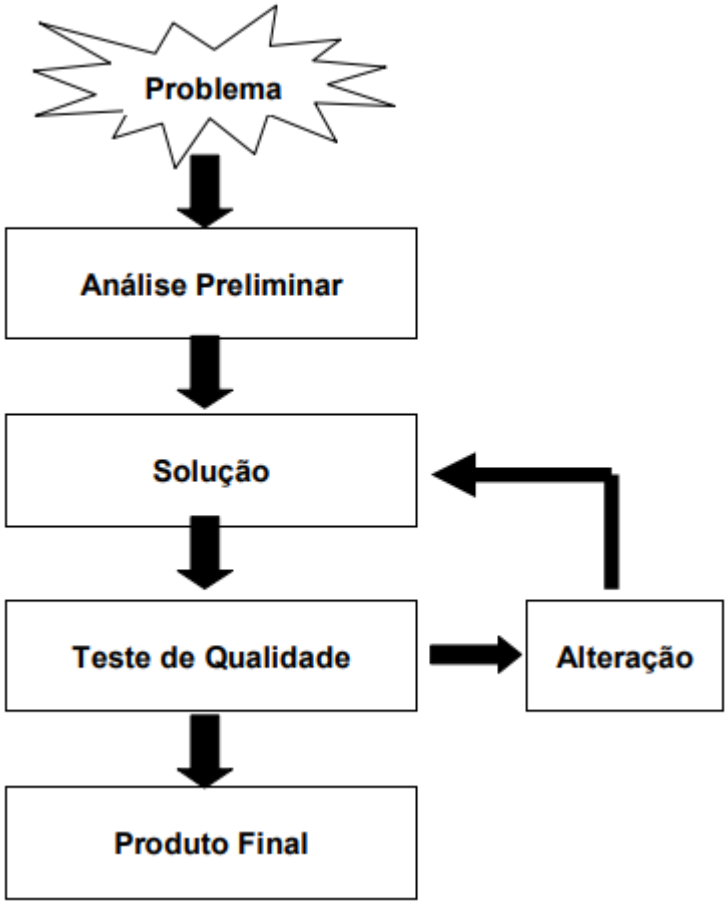
1 **Definição ->** Deve descrever exatamente quais são as instruções que devem ser executadas e em que sequência. Deve ser tornado explícito o maior número possível de informações, pois a falta de alguma informação pode levar a uma interpretação errada do algoritmo;
2 **Ausência de Ambiguidade ->** Não deve deixar dúvidas sobre o que deve ser feito. A ambiguidade acerca do que deve ser feito também pode levar a uma interpretação errada do algoritmo;
3 **Eficácia ->** Conseguir resolver o problema em qualquer situação. Todas as situações de exceção que possam alterar o comportamento do algoritmo devem ser especificadas e tratadas;
4 **Eficiência ->** Resolver o problema com o mínimo de recursos. Sempre se deve buscar aquele algoritmo que, dentre os diversos algoritmos que resolvam um mesmo problema, utilize a menor quantidade de recursos. No caso de algoritmos para processamento de dados, os recursos a serem considerados são espaços de memória (principal e auxiliar) e tempo de processamento (economia de C.P.U.), entre outros.

Estratégias na Construção de Algoritmos:

- Especifique o problema claramente e entenda-o completamente;
- Explicite todos os detalhes supérfluos;
- Entre no problema (envolva-se totalmente com o problema);
- Use todas as informações disponíveis;
- Decomponha o problema (Top-Down);
- Use o sentido inverso, se necessário (Bottom-Up).

Como Construir Algoritmos?

Visão Esquemática da Construção de Algoritmos:



- **Análise Preliminar ->** Entenda o problema com a maior precisão possível, identifique os dados; identifique os resultados desejados. **Solução ->**Desenvolva um algoritmo para resolver o problema.
- **Teste de Qualidade ->** Execute o algoritmo desenvolvido com dados para os quais o resultado seja conhecido. O ideal é que o universo dos dados tenha todas as combinações possíveis.

Note que a qualidade de um algoritmo pode ser limitada por fatores como tempo para a sua confecção e recursos disponíveis.

- **Alteração ->** Se o resultado do teste de qualidade não for satisfatório, altere o algoritmo e submeta-o a um novo teste de qualidade.
- **Produto Final ->** O algoritmo concluído e testado, pronto para ser aplicado.