Projeto de programas

Marco A L Barbosa
malbarbo.pro.br

Departamento de Informática
Universidade Estadual de Maringá

O André viaja muito. Sempre antes de fazer uma viagem ele calcula o quanto ele irá gastar com combustível. Ele determina a distância que ele irá percorrer na viagem, o preço do litro do combustível e consulta as suas anotações para ver o consumo do carro, isto é, a quantidade de quilômetros que o carro anda com um litro de combustível e então faz o cálculo do custo. O André acha um pouco chato fazer os cálculos na mão, então ele pediu para você escrever um programa que faça os cálculos para ele.

Como projetar um programa que atenda a necessidade do André?

Seguindo um processo, uma sequência de etapas.

Projetar programas que funcionem corretamente e sejam bem escritos é um desafio, seguir um processo é uma ferramenta indispensável nesse processo.

No início, para problemas simples, o processo poderá parecer muito custoso, mas vamos apreciar a sua utilidade conforme progredimos.

O processo que vamos seguir está dividido em 6 etapas:

- Análise
- Definição dos tipos de dados
- Especificação
- Implementação
- Verificação
- Revisão

Ao final de cada etapa produzimos resultados que serão utilizados nas etapas posteriores, então devemos tentar seguir as etapas em ordem. No entanto, em algumas situações, pode ser necessário mudar a ordem.

Por exemplo, às vezes estamos na implementação e encontramos uma nova condição e devemos voltar e alterar a especificação. Em outra situação não estamos conseguindo entender o problema (análise) e então fazemos alguns exemplos (especificação) para nos ajudar.

Mas devemos evitar fazer a implementação diretamente!

O André viaja muito. Sempre antes de fazer uma viagem ele calcula o quanto ele irá gastar com combustível. Ele determina a distância que ele irá percorrer na viagem, o preço do litro do combustível e consulta as suas anotações para ver o consumo do carro, isto é, a quantidade de quilômetros que o carro anda com um litro de combustível e então faz o cálculo do custo. O André acha um pouco chato fazer os cálculos na mão, então ele pediu para você escrever um programa que faça os cálculos para ele.

Objetivo: determinar o que deve ser feito.

- Quais informações são relevantes e quais podem ser descartadas?
- Existe alguma omissão?
- Existe alguma ambiguidade?
- Quais conhecimentos do domínio do problema são necessários?

Resultado

Calcular o custo em reais para percorrer uma determinada distância levando em consideração o desempenho do carro e o preço do litro do combustível.

Análise

Calcular o custo em reais para percorrer uma determinada distância levando em consideração o desempenho do carro e o preço do litro do combustível.

Objetivo: definir como as informações serão representadas como dados no programa.

- Quais são as informações envolvidas no problema?
- Como as informações serão representadas?

Resultado

As informações são a distância em Km, rendimento em Km/l, preço em R$/l e o custo da viagem em R$.

Todos os valores serão representados por números positivos.

Análise

Calcular o custo em reais para percorrer uma determinada distância levando em consideração o desempenho do carro e o preço do litro do combustível.

Tipos de dados

As informações são a distância em Km, rendimento em Km/l, preço em R$/l e o custo da viagem em R$.

Todos os valores serão representados por números positivos.

Objetivo: especificar com mais precisão e com exemplos o que o programa deve fazer.

- Assinatura da função (nome, tipo das entradas e saídas)
- Propósito da função
- Exemplos de entrada e saída

Assinatura e propósito da função

```
// Calcula o custo em reais para percorrer a distancia especificada
// considerando o rendimento do carro e o preco do litro do combustível.
double custo_viagem(double distancia, double rendimento, double preco)
{
    return 0.0;
}
```

Exemplos

Ilustrar com exemplos de entrada e saída o funcionamento da função.

Como escolher bons exemplos?

- Usar valores de casos práticos para o problema
- Considerar diversas situações, incluindo casos extremos

```
// custo_viagem(120, 10, 5)
// (120 / 10) * 5 -> 12 * 5 -> 60
// custo_viagem(300, 15, 6)
// (300 / 15) * 6 -> 20 * 6 -> 120
```

```c
// Calcula o custo em reais para percorrer
// a distancia especificada considerando o
// rendimento do carro e o preco do litro
// do combustível.
// Exemplos
// custo_viagem(120, 10, 5)
// (120 / 10) * 5 -> 12 * 5 -> 60
// custo_viagem(300, 15, 6)
// (300 / 15) * 6 -> 20 * 6 -> 120
double custo_viagem(double distancia,
                    double rendimento,
                    double preco)
{
    return 0.0;
}
```

Objetivo: escrever o corpo da função para que ela faça o que está na especificação.

Baseado nos exemplos, generalizamos a forma de calcular a resposta.

```c
double custo_viagem(double distancia,
                    double rendimento,
                    double preco)
{
    return (distancia / rendimento)
           * preco;
}
```

Verificação

Objetivo: verificar se a implementação está de acordo com a especificação.

Usamos os exemplos para testar se o programa funciona corretamente.

Se o programa produzir a resposta errada para algum exemplo, onde está o erro?

- No exemplo
- No código da função
- Em ambos

Primeiro verificamos os exemplos, se algum estiver errado, corrigimos o exemplo e executamos novamente os testes.

Se os exemplos estiverem corretos, então analisamos o corpo da função para tentar identificar e corrigir o erro. Após a alteração do código, executamos novamente os testes.

Objetivo: alterar a organização do programa para que fique mais fácil de ser lido, entendido e alterado.

Se modificarmos o código, precisamos executar novamente os testes!

Alguma parte desse processo parece repetitiva?

Sim, a execução dos exemplos e a verificação se as saídas estão corretas.

Ao invés de executarmos cada exemplo manualmente, vamos usar uma biblioteca que executa os exemplos e verifica se as saídas estão corretas automaticamente!

A biblioteca que vamos utilizar foi desenvolvida especificamente para esta disciplina e chama "Begin Student C++" (bscpp).

Para usar a biblioteca é necessário fazer o download do arquivo `bscpp.hpp` (https://malbarbo.pro.br/bscpp.hpp) e salvá-lo no mesmo diretório dos arquivos `.cpp`.

Incluímos a biblioteca com a instrução

```
#include "bscpp.hpp"
```

Escrevemos os exemplos dentro de um bloco `examples` após a função. Cada exemplo é especificado por uma cláusula `check_expect`.

No `main` chamamos a função `run_tests`.

Para compilar o programa, precisamos especificar a opção `-std=c++17` pois a biblioteca de testes foi escrita usando o C++17.

Testes automatizados

```
#include "bscpp.hpp"

// Calcula o custo em reais para percorrer a distancia especificada
// considerando o rendimento do carro e o preco do litro do combustível.
double custo_viagem(double distancia, double rendimento, double preco) {
    return (distancia / rendimento) * preco;
}

examples {
    check_expect(custo_viagem(120, 10, 5), 60);
    check_expect(custo_viagem(300, 15, 6), 120);
}

int main() {
    run_tests();
}
```

Compilação

```
> g++ -std=c++17 -o programa programa.cpp
```

Execução dos testes

```
> ./programa
Ran 2 tests.
All tests passed!
```

Exemplo

Vamos fazer outro exemplo.

Um construtor precisa calcular a quantidade de azulejos necessários pra azulejar uma determinada parede. Cada azulejo é quadrado e tem 20cm de lado. Ajude o construtor e defina uma função que receba como entrada o comprimento e a altura em metros de uma parede e calcule a quantidade de azulejos inteiros necessários para azulejar a parede. Considere que o construtor nunca perde um azulejo e que recortes de azulejos não são reaproveitados.

Objetivo: determinar o que deve ser feito.

- Quais informações são relevantes e quais podem ser descartadas?
- Existe alguma omissão?
- Existe alguma ambiguidade?
- Quais conhecimentos do domínio do problema são necessários?

Resultado

Calcular o número de azulejos necessários para azulejar uma parede com determinado comprimento e altura, considerando que o azulejo mede 0,2,0m x 0,2m e que nenhum azulejo é perdido e que recortes são descartados.

Análise

Calcular o número de azulejos necessários para azulejar uma parede com determinado comprimento e altura, considerando que cada azulejo mede 0,2m x 0,2m e que nenhum azulejo é perdido e que recortes são descartados.

Objetivo: definir como as informações serão representadas como dados no programa.

- Quais são as informações envolvidas no problema?
- Como as informações serão representadas?

Resultado

O comprimento e a altura da parede são dados em metros e representados com números positivos.

O número de azulejos é representado por um número inteiro positivo.

Análise

Calcular o número de azulejos necessários para azulejar uma parede com determinado comprimento e altura, considerando que cada azulejo mede 0,2m x 0,2m e que nenhum azulejo é perdido e que recortes são descartados.

Tipos de dados

O comprimento e a altura da parede são dados em metros e representados com números positivos.

O número de azulejos é representado por um número inteiro positivo.

Objetivo: especificar com mais precisão e com exemplos o que o programa deve fazer.

- Assinatura da função (nome, tipo das entradas e saídas)
- Propósito da função
- Exemplos de entrada e saída

```c
// Calcula o número de azulejos de 0,2mx0,2m necessários para azulejar uma area
// de tamanho comprimento x altura (em metros) considerando que nenhum azulejo
// é perdido e que recortes são descartados.
int numero_azulejos(double comprimento, double altura)
{
    return 0;
}
```

Exemplos (projeto feito em sala)

```
examples
{
   // alguns exemplos práticos
   // ceil(2.0 / 0.2) * ceil(2.4 / 0.2) -> 10 * 12 -> 120
   check_expect(numero_azulejos(2.0, 2.4), 120);

   check_expect(numero_azulejos(1.5, 2.3), 96);
   // ceil(1.5 / 0.2) * ceil(2.3 / 0.2) -> 8 * 12 = 96

   // alguns casos extremos
   check_expect(numero_azulejos(0.2, 0.2), 1);
   check_expect(numero_azulejos(0.3, 0.2), 2);
   check_expect(numero_azulejos(0.3, 0.3), 4);
   check_expect(numero_azulejos(0.4, 0.4), 4);
}
```

Implementação

```
int numero_azulejos(double comprimento, double altura) {
    return ceil(comprimeto / 0.2) * ceil(altura / 0.2);
}
```

Verificação

```
Ran 6 tests.
All tests passed!
```

Revisão

O código está ok.

Valor máximo

Projete uma função que encontre o maior valor entre dois números dados.

Análise e definição de tipos de dados

Análise

- Encontrar o valor máximo entre dois número dados

Tipos de dados

- Os valores serão números inteiros

Especificação

```c
// Encontra o valor máximo entre a e b.
int maximo(int a, int b)
{
    return 0;
}

examples
{
    check_expect(maximo(20, 10), 20);
    check_expect(maximo(5, 10), 10);
    check_expect(maximo(5, 5), 5);
}
```

Implementação

Como determinar o valor máximo entre dois números utilizando as operações que vimos até agora?

Não tem como!

A resposta depende de uma condição, se **a** > **b**, então a resposta é **a**, senão a reposta é **b**.

Vamos ver como expressar esse tipo de construção a seguir.