Análise Estrutural do Código C para Verificação de Número Primo

a) Qual o grafo de fluxo de controle para este código?

O grafo de fluxo de controle representa os diferentes caminhos que podem ser seguidos pela execução do código.

1. Início
2. Verifica se `num <= 1`
   - Sim: Vai para o 3
   - Não: Vai para o 4
3. Retorna `false` e termina
4. Laço `for (int i = 2; i * i <= num; i++)`
   - Se a condição é falsa, vai para o 7
   - Se a condição é verdadeira, vai para o 5
5. Verifica se `num % i == 0`
   - Sim: Vai para o 6
   - Não: Volta para o 4 (próxima iteração do laço)
6. Retorna `false` e termina
7. Retorna `true` e termina
8. Verifica se o resultado de `ehPrimo` é `true`
   - Sim: Imprime "é primo" e termina
   - Não: Imprime "não é primo" e termina

b) Quantos caminhos independentes existem neste código?

Os caminhos independentes são aqueles que não podem ser representados como combinações lineares de outros caminhos. Para este código, existem 3 caminhos independentes principais:

1. Quando o número é menor ou igual a 1.
2. Quando o número é primo.
3. Quando o número não é primo e encontra um divisor dentro do laço `for`.

c) Liste todos os caminhos independentes identificados.

Os caminhos independentes são:

1. Caminho 1:
   - 1 → 2 → 3

2. Caminho 2:
   - 1 → 2 → 4 → 7 → 8

3. Caminho 3:
   - 1 → 2 → 4 → 5 → 6 → 8

d) Para cada caminho independente, descreva um caso de teste que garantiria a cobertura desse caminho.

1. Caminho 1:
   - Caso de teste: `num = -1` ou `num = 0` ou `num = 1`
   - Esperado: "não é primo"

2. Caminho 2:
   - Caso de teste: `num = 29`
   - Esperado: "é primo"

3. Caminho 3:
   - Caso de teste: `num = 30`
   - Esperado: "não é primo"

e) Quais são as condições lógicas presentes no código?

As condições lógicas no código são:

1. `num <= 1`
2. `i * i <= num` (condição do laço `for`)
3. `num % i == 0` (condição dentro do laço `for`)

f) Descreva um conjunto mínimo de casos de teste que garantam a cobertura de todas as condições lógicas.

Para cobrir todas as condições lógicas, os seguintes casos de teste são necessários:

1. `num = -1` (cobre `num <= 1`)
2. `num = 1` (cobre `num <= 1`)
3. `num = 2` (cobre o laço não sendo executado `i * i > num`)
4. `num = 4` (cobre `num % i == 0` sendo `true`)
5. `num = 29` (cobre `i * i <= num` com múltiplas iterações do laço e `num % i == 0` sendo `false`)

g) Descreva os casos de teste usando análise de valor limite considerando que um número primo é aquele que é maior que 1 e divisível apenas por 1 e por ele mesmo.

1. `num = 1`
   - Limite inferior para números não primos.
   - Esperado: "não é primo"

2. `num = 2`
   - Limite inferior para números primos.
   - Esperado: "é primo"

3. `num = 3`
   - Próximo número primo.
   - Esperado: "é primo"

4. `num = 4`
   - Primeiro número não primo maior que 2.
   - Esperado: "não é primo"

5. `num = 29`
   - Verificação para um número primo maior.
   - Esperado: "é primo"

6. `num = 30`
   - Verificação para um número não primo maior.
   - Esperado: "não é primo"