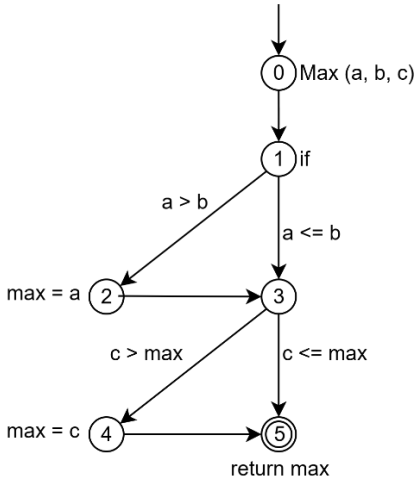
CÁC TIÊU CHÍ BAO PHỦ MÃ NGUỒN

III. Tiêu chí bao phủ nhánh – Decision/Edge/Branch Coverage

Thiết kế Testcase sao cho đảm bảo các nhánh của Node quyết định được thực hiện ít nhất 1 lần (Chọn lộ trình khả thi tối thiểu bao phủ tất cả các nhánh của Node quyết định trong đồ thị)

Ví dụ 1: Hàm tìm max

```
int Max (int a, int b, int c){
    int max;
    if(a > b)
        max = a;
    if(c > max)
        max = c;
    return max;
}
```



Paths = {012345, 0135, 01345, 01235}

Theo tiêu chí 2: lộ trình 012345 và 0135 là lộ trình khả thi tối thiểu bao phủ được các nhánh của đỉnh quyết định trong CFG

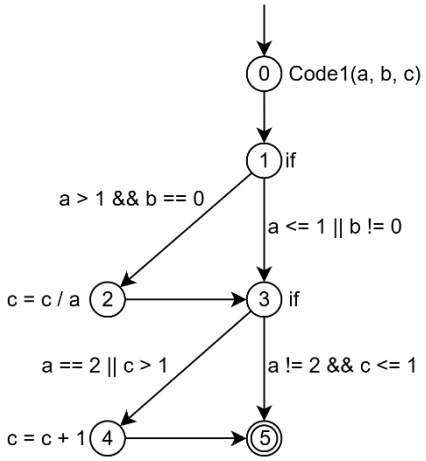
Giải thích:

Lộ trình 012345	**Data test**
a > b max = a c > max max = c return max ➔ c > a > b	a = 5 b = 2 c = 10
Lộ trình khả thi	
Lộ trình 0135	**Data test**
a <= b c <= max return max ➔ b >= a và max >= c	a = 5 b = 5 c = 1

Lộ trình khả thi

Ví dụ 2:

```
void Code1 (int a, int b, int c){
    if((a > 1) && (b == 0))
        x = x / a;
    if((a == 2) || (x > 1))
        x = x + 1;
}
```



Paths = {012345, 0135, 01345, 01235}

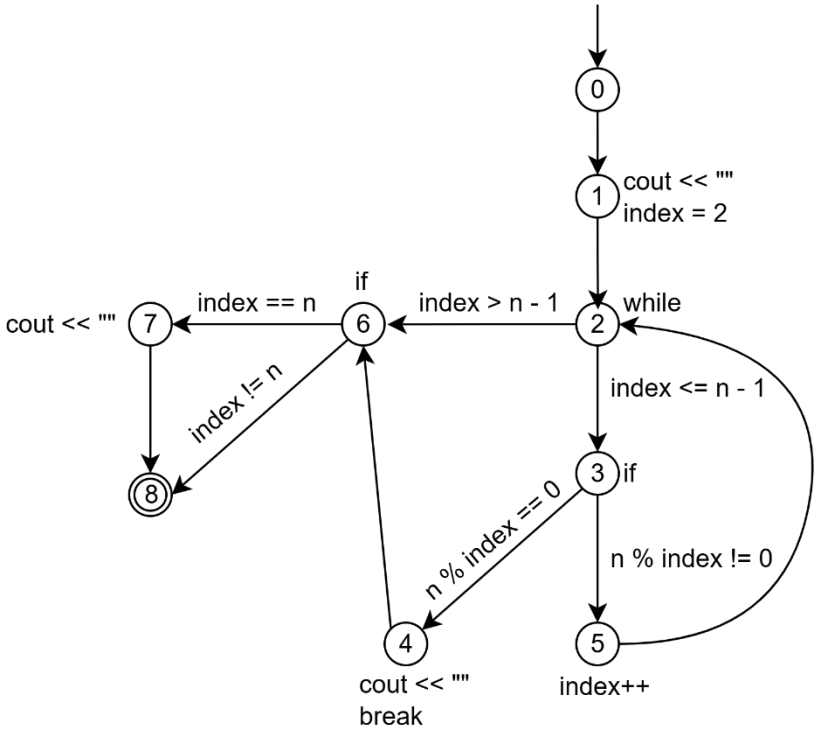
Theo tiêu chí 2: lộ trình 012345 và 0135 là lộ trình khả thi tối thiểu bao phủ được các nhánh của đỉnh quyết định trong CFG

Giải thích:

Lộ trình 012345	**Data test**
a > 1 && b == 0 c = c / a a == 2 || c > 1 c = c + 1	a = 2, b = 0, c = 10 c = 10 / 2 = 5 2 == 2 || 5 > 1 c = 5 + 1
Lộ trình khả thi	
Lộ trình 0135	**Data test**
a <= 1 || b != 0 a != 2 && c <= 1	a = 1 b = 5 c = 1
Lộ trình khả thi	

Ví dụ 3:

```cpp
int main(){
    int n, index;
    cout << "Nhap so n: " << n;
    index = 2;
    while(index <= n - 1){
        if(n % index == 0){
            cout << "Khong phai la so nguyen to";
            break;
        }
        index++;
    }
    if(index == n)
        cout << "La so nguyen to";
}
```



Paths = {012678, 01268, 01234678, 0123468, 01(235)*2678, 01(235)*268, 01(235)*234678, 01(235)*23468, …}

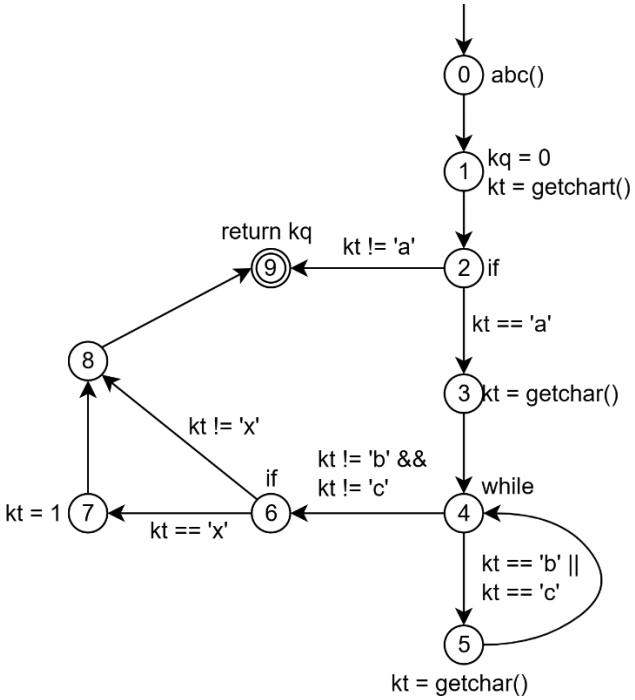
Theo tiêu chí 2: lộ trình 012352678 và 0123468 là lộ trình khả thi tối thiểu bao phủ được các nhánh của đỉnh quyết định trong CFG

Giải thích:

Lộ trình 012352678	**Data test**
index = 2 index <= n – 1 n % index != 0 index++ index > n – 1 index == n cout <<””	n = 3 2 <= 3 – 1 3 % 2 != 0 2++ = 3 3 > 3 – 1 3 == 3
Lộ trình khả thi	
Lộ trình 0123468	**Data test**
index = 2 index <= n – 1 n % index == 0 cout << “” break index != n	n = 4 2 <= 4 – 1 4 % 2 == 0 2 != 4
Lộ trình khả thi	

Ví dụ 4:

```
int abc(){
    char kt;
    int kq = 0;
    kt = getchar();
    if(kt == 'a'){
        kt = getchar();
        while((kt =='b') || (kt == 'c'))
            kt = getchar();
        if(kt == 'x') kq = 1;
    }
    return kq;
}
```



s

Paths = {0129, 012346789, 0123(45)*46789, 01234689, 0123(45)*689,…}

Theo tiêu chí 2: lộ trình 0129, 012346789 và 0123454689 là lộ trình khả thi tối thiểu bao phủ được các nhánh của đỉnh quyết định trong CFG

<u>Giải thích</u>:

Lộ trình 0129	**Data test**
kq = 0 kt = getchar() kt != ‘a’ return kq	kt = ‘d’ ‘d’ != ‘a’ return kq = 0
Lộ trình khả thi	
Lộ trình 012346789	**Data test**
kq = 0 kt = getchar() kt == ‘a’ kt = getchar() kt != ‘b’ && kt != ‘c’ kt == ‘x’ kt = 1 return kq	 kt = ‘a’ ‘a’ == ‘a’ kt = ‘x’ ‘x’ != && ‘x’ != ‘c’ ‘x’ == ‘x’ return kq = 1

Lộ trình khả thi	
Lộ trình 0123454689	**Data test**
kq = 0 kt = getchar() kt == ‘a’ kt = getchar() kt == ‘b’ || kt == ‘c’ kt = getchar() kt != ‘b’ && kt != ‘c’ kt != ‘x’ return kq	 kt = ‘a’ ‘a’ == ‘a’ kt = ‘b’ ‘b’ == ‘b’ kt = ‘f’ ‘f’ != ‘b’ && ‘f’ != ‘c’ ‘f’ != ‘x’ return kq = 0