Tutorial Link https://codequotient.com/tutorials/Short-Circuit of operators in C/5a22a0b6c66cfe38f2962257

TUTORIAL

Short-Circuit of operators in C

Chapter

1. Short-Circuit of operators in C

In C we can use logical operators to combine multiple conditions in if blocks. For example to check a number is greater than 1 and less than 20 we can use,

```
#include <stdio.h>

int main()
{
 int num = 29;
 if ( num > 1 && num < 20)
   printf("num is between 1 and 20. \n");
 else
   printf("num is not between 1 and 20. \n");
 return 0;
}
```

But the logical AND and OR operators also called short-circuit operators. Due to their characteristic of non-evaluating the second condition if result is predictable. In other words, if first condition to AND operator is false, no matter the what the second condition evaluates to, the overall output is always false, as AND requires both inputs to be true to produce a true as output. Similarly, if first condition to OR operator is true, no matter the what the second condition

evaluates to, the overall output is always true, as OR requires at least one inputs to be true to produce a true as output. So, when we combine the conditions with logical operators, it is not guaranteed that both conditions were checked.

This situation cause not obvious outputs sometimes. For example,

```c
#include <stdio.h>

int main()
{
  int num = 29;
  char ch='C';

  // This never assign 'A' to ch, as first condition evalautes to false in AND
  if (num < 10 && (ch = 'A'))
    printf("It will not print as condition is false. \n");

  printf("num = %d, ch = %c \n",num, ch);

  // Also, this never assign 'A' to ch, as first condition evalautes to true in OR
  if (num > 10 || (ch = 'A'))
    printf("It will print as condition is true. \n");

  printf("num = %d, ch = %c \n",num, ch);

  return 0;
}

```

So, if the second condition have some side-effect just like above, they will never be committed. So we have to take care while writing the conditions using logical operators.

codequotient

tient.com

Tutorial by codequotient.com | All rights reserved, CodeQuotient 2023