APS105 Lecture 7 Notes

Last lecture: if-else statements, relational operators, logical operators, lazy evaluation

Today: More complex relational and logical expressions, example code on how to know a cher falls in alphabets, De Morgan's Law, dangling else problem and nested if-statements.

Recall:

ASCII code of characters is ordered as follows:
'0' < '1' < '2' < '3' ---- '9' <
'A' < 'B' < 'C' < ---- 'Z' < 'a' < 'b' ---- < 'z'

ASCII code is the character encoding for characters, e.g. 'A' is encoded as $(0100\ 0001)_2$ => 65 in decimal

Write a C program that takes a character from the user, and output if this character is a number between '0' and '9'.

Scanned with CamScanner

```
char charInput;
printf("Enter char: ");
scanf("%c", &charInput);
if ( charInput >= '0' && charInput <= '9' )
        printf("You entered a number between 0 and 9");
else
        printf("You didn't enter a number");
```

Is there another way to write the expression?

```
charInput >= '0' && charInput <= '9'
```

Yes, we can switch the individual relational conditions and Not them.

Recall: `if (done == false)` is equivalent to `if (!done)`

So, `charInput < '0'` is opposite of `charInput >= '0'`

then `!(charInput < '0')` is equivalent to `(charInput >= '0')`

Also, `!(charInput > '9')` is equivalent to `(charInput <= '9')`

Scanned with CamScanner

Our expression can now be written as

```
!(charInput < '0') && !(charInput > '9')
```

According to De Morgan's law, if A and B are booleans

I $!(A \;\&\&\; B) = !A \;||\; !B$

The NOT enters the bracket to individual components and switches && to ||

II $!(A \;||\; B) = !A \;\&\&\; !B$

The NOT enters the bracket to individual components and switches || to &&

Back to our example, using Law II,

```
!(charInput < '0') && !(charInput > '9')
```

is equivalent to

```
!(charInput < '0' || charInput < '9')
```

Scanned with CamScanner

Precedence Rule (of all operators discussed so far)

- Recap
- Relational operators
- Logical or boolean operators
- NEW sizeof()

(type) () ! sizeof()

* / %

\+ -

< <= > >=

relational operators: == !=

&&

||

assignment operators: = += -= *= /= %= (right to left)

e.g. i = j = k = l

① k = l

② j = (k = l)

③ i = (j = (k = l))

```
int x = 5;
if( ! x > 5 )
```

E.g. ① ! x ② x > 5

! x ⇒ false

false ⇒ 0

0 > 5 ⇒ false

But correct way is, if (! (x > 5))

① x > 5

② ! (true)

x > 5 ⇒ true

! (true) ⇒ false

Scanned with CamScanner

sizeof (int) ⇒ 4

sizeof (double) ⇒ 8

sizeof (char) ⇒ 1

sizeof() → gets the size of a type, not a function as it takes in type not variable.

Nested if -statements

```
if(   ){

}else {
   if(   ){

   }else{
      if(   ){

      }
      else {

      }
}
```

⟹ Elegant & more elegant

```
if(   ){

}else if(   ){

}else if(   ){

}else {

}
```

Scanned with CamScanner

Find maximum of 3 ints x, y, z using if-statements

Recall: with 2 ints

```
if (x > y)
    max = x;
else
    max = y;
```

Now, we have 3 ints

① Get the maximum of 2 ints

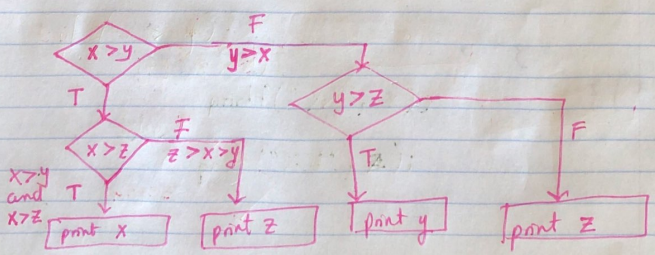
② Compare it with the 3rd int

Because if $x > y$, then no way y will be max.

e.g. 1 $x = 3$, $y = 7$, $z = 10$;

(i) Check x & y ⇒ y is max.

(ii) Check y & z ⇒ z is max.



Scanned with CamScanner

```
if (x>y)
    if (x>z)
        printf("%d", x);
    else
        printf("%d", z);
else
    if (y>z)
        printf("%d", y);
    else
        printf("%d", z);
```

① x > y & x > z

② x > y & z > x

① y > x & y > z

② y > x & z > y

Dangling if-else problem

```
if ( condition )
    if ( condition )
        statements;
else
    statements;
```

To wish if does else belong?

else always belongs to the nearest if

Scanned with CamScanner