שיעור 2

תוכן

- Logical operators

http://www.learncpp.com/cpp-tutorial/36-logical-operators/

Logical OR (operator ||)		
Left operand	Right operand	Result
false	false	false
false	true	true
true	false	true
true	true	true

Logical AND (operator &&)		
Left operand	Right operand	Result
false	false	false
false	true	false
true	false	false
true	true	true

קדימות - חשוב לשים לב שלאופרטור && יש קדימות על האופרטור ||

הביטוי הבא:

```
nValue1 || nValue2 && nValue3
```

יתורגם ל:

```
nValue1 || (nValue2 && nValue3)
```

ולא ל:

```
(nValue1 || nValue2) && nValue3.
```

לכן כדי למנע בלבול, כדאי להשתמש בסוגריים

חוקי דה מורגן

חוקי דה מורגן הם חוקים בתחום הלוגיקה המתמטית שקובעים כי:

`!(x && y)` is equivalent to `!x || !y`
`!(x || y)` is equivalent to `!x && !y`

לכן חשוב שתשימו לב שהביטוי הבא **לא** נכון:

`!(x && y)` is the same thing as `!x && !y`

ASIDE

מה היא מערכת המספרים הבינארית?

10101 base 2

1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 21

101010 base 2

1*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 42

שימו לב שכרגע ביצענו כפל ב-2

123 base 10

1*10^2 + 2*10^1 + 3*10^0 = 123

- Bitwise operators

http://www.learncpp.com/cpp-tutorial/38-bitwise-operators/

ב-CPP קיימים שישה אופרטורים שמאפשרים לתפעל ביטים

תיאור הפעולה	שימוש	סימון	אופרטור
להזיז את הביטים ב-x y מקומות שמאלה	x<<y	<<	להזיז ביטים שמאלה
להזיז את הביטים ב-x y מקומות ימינה	x>>y	>>	להזיז ביטים ימינה
להפוך את כל הביטים ב-x	~x	~	שלילה
פעולת וגם על הביטים ב-x וב-y	x&y	&	וגם
...	x|y	|	או
...	x^y	^	או אקסקלוסיבי

<u>חוק</u> - בזמן שימוש באופרטורים של ביטים חובה להשתמש במשתנים שהם unsigned (כלומר שאין להם ייצוג שלילי)

<u><<</u>
הייצוג הבינארי של 3 הוא 0011
(3<<1) 0110 = 6
(3<<2) 1100 = 12
(3<<3) 1000 = 8

שימו לב שנעלם לנו ביט כשהזזנו את 0011 3 מקומות שמאלה

<u>>></u>

(6>>1) 0011 = 3

<u>~</u>

1011 הופך ל 0100

<u>&</u>

1011
&
0001

0001

↓

1101
|
0100

1101

```
  ^
 1101
  ^
 0100
------
 1001
```

- pre increment / post increment

```
int j = i++; // j will contain i, i will be incremented.
```

Vs:

```
int j = ++i; // i will be incremented, and j will contain i+1.
```