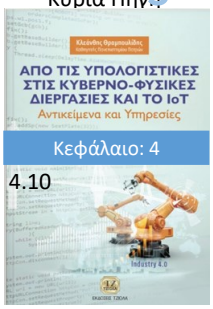
Αντικειμενοστρεφής Προγραμματισμός (Object-Oriented Programming)

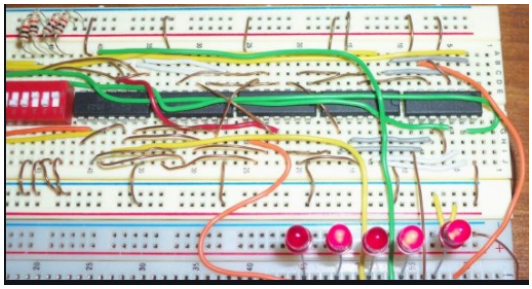
(CEID_NNY106)

LEGO Approach - Activity 4.8

Logic Gates Circuit Simulator

Κύρια Πηγή:





Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
https://sites.google.com/site/thramboulidiskleanthis/

Java

High-level programming language

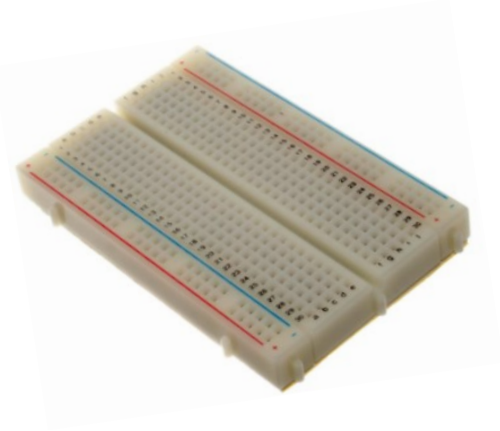
Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. Wikipedia

Designed by: James Gosling

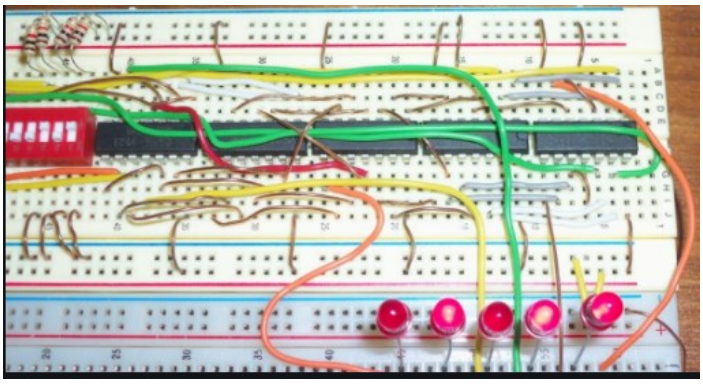
First appeared: May 23, 1995; 27 years ago

Paradigm: Multi-paradigm: generic, object-oriented (class-based), functional, imperative, reflective, concurrent

Breadboard – Logic Gates Circuit

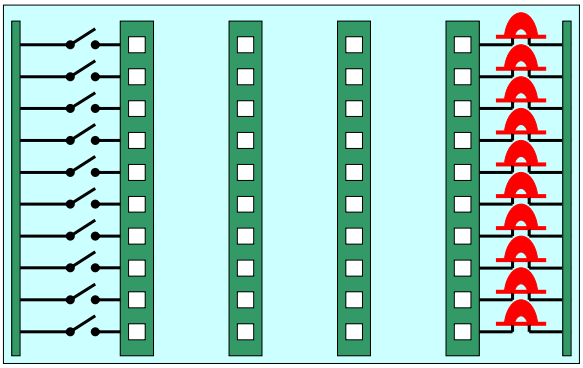


Calculator with logic gates



Εξομοιωτής κυκλωμάτων λογικών πυλών

- Στα πλαίσια της άσκησης αυτής θα αναπτύξετε ένα πρόγραμμα σύμφωνα με το οποίο το σύστημα θα σας επιτρέπει
  - να δημιουργείτε ένα κύκλωμα πάνω σε ένα breadboard,
  - να δίνετε τιμές στις εισόδους του κυκλώματος
  - να παίρνετε τις τιμές των εξόδων του κυκλώματος σας.



Καθώς **δεν είστε σε θέση να αναπτύξετε ένα τέτοιο πρόγραμμα** θα πρέπει να εκτελέσετε ένα σύνολο από δραστηριότητες και ενέργειες που θα σας οδηγήσουν βήμα προς βήμα στη δημιουργία του δικού σας εξομοιωτή κυκλωμάτων λογικών πυλών.

Στόχος της άσκησης

Προσέξτε όμως! στόχος της άσκησης δεν είναι να αναπτύξετε το πρόγραμμα με τον τρόπο που ήδη γνωρίζετε.

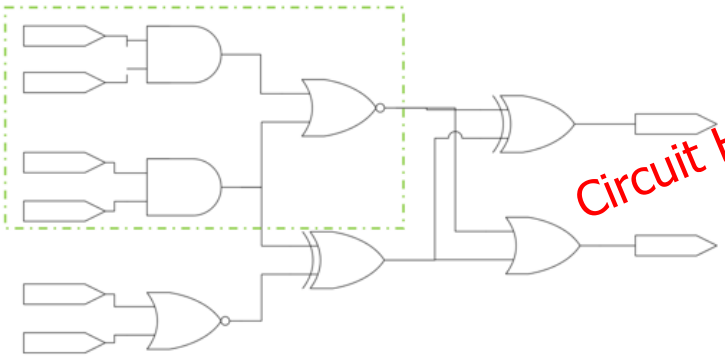
- Ο κυρίαρχος στόχος της άσκησης είναι

η αναπαράσταση και οργάνωση των δεδομένων ενός φυσικού προβλήματος.

Iteration No 1

3.1. Λειτουργικότητα του Iteration

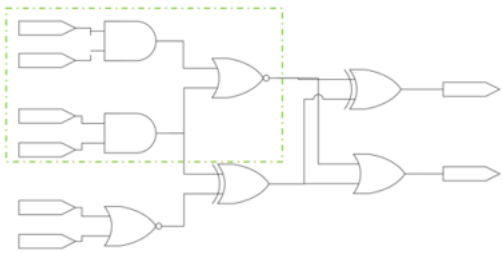
Αναπτύξτε μια εφαρμογή **Logic Gates Circuit Simulator** που θα υπολογίζει την έξοδο του κυκλώματος λογικών πυλών που δίνει το **Σχήμα 1**. Το **Σχήμα 2** δίνει τα σύμβολα χαρακτηριστικών τύπων λογικών πυλών.



Σχήμα 1 *Απλό κύκλωμα λογικών πυλών.*

Αναγνώριση βασικών Διεργασιών/Υπηρεσιών

- Εντοπίστε τις βασικές διεργασίες/υπηρεσίες και ορίστε κατάλληλες συναρτήσεις /μεθόδους που θα τις υλοποιούν



`defineCircuitInputs()`

`calculateOutputs()`

`printCircuitOutputs()`

`printCircuitStatus()`

`printCircuit()`

Iteration No 1

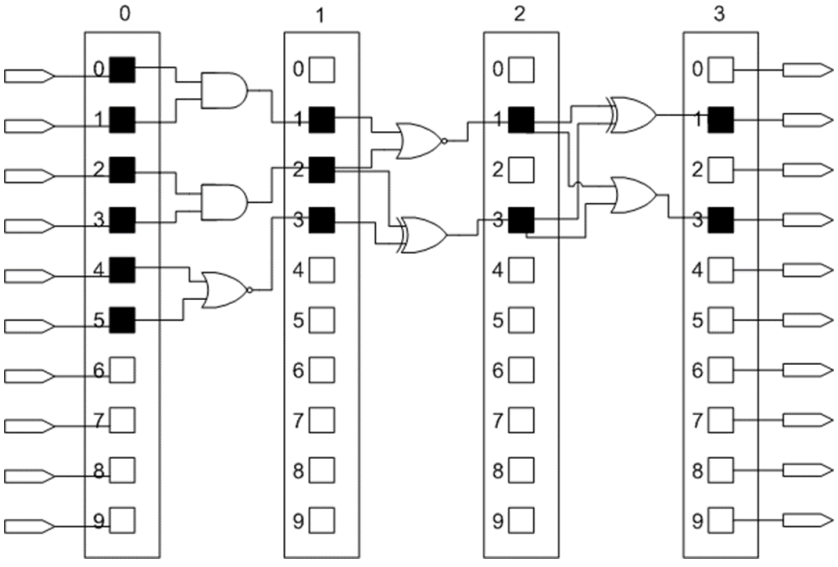
3.2. Βήματα εκτέλεσης του Iteration

E1.1. Δώστε ένα πρόχειρο διάγραμμα κλάσεων θεωρώντας το κύκλωμα ως ένα σύνολο από στιγμιότυπα τα οποία αλληλοεπιδρούν μεταξύ τους.

E1.2. Δώστε τον κώδικα που υλοποιεί σε πρώτη φάση το τμήμα του κυκλώματος που βρίσκετε μέσα στο ορθογώνιο πλαίσιο. Ελέγξτε αν το πρόγραμμα σας λειτουργεί σωστά.

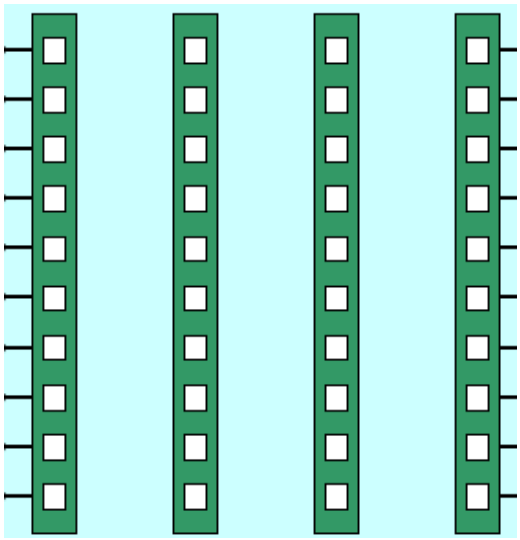
E1.3. Προχωρήστε στην υλοποίηση του προγράμματος για όλο το κύκλωμα. Ελέγξτε αν το πρόγραμμα σας λειτουργεί σωστά.

Iteration No 2 – Το κύκλωμα πάνω σε breadBoard



Αναπαράσταση breadboard

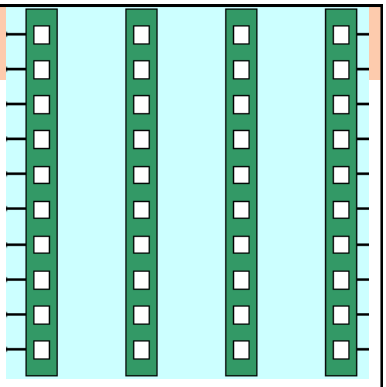
- **Αναπαράσταση του breadboard**
  - Κάθε υποδοχή (**pin**) μπορεί να έχει δύο τιμές High ή Low (0 ή 1, 0 V ή 5 V).
  - Τι τύπου μεταβλητή θα χρησιμοποιήσετε για την αναπαράσταση της υποδοχής;
  - Πόσες μεταβλητές θα χρησιμοποιήσετε για την αναπαράσταση όλων των υποδοχών της πλακέτας;



Αναπαράσταση breadboard

- **Αναπαράσταση του breadboard**

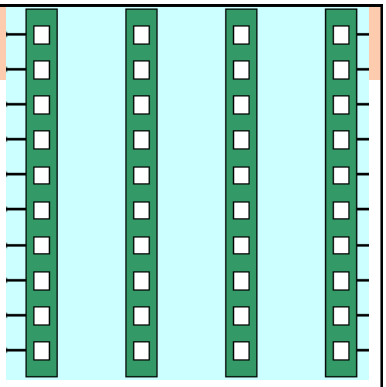
```
boolean inputColumnPin1;
boolean inputColumnPin2;
…
boolean inputColumnPin10;
```



```
boolean inputColumn[10];
boolean middleColumn1[10];
boolean middleColumn2[10];
boolean outputColumn[10];
```

```
boolean breadBoard[4][10];
```

Αναγνωσιμότητα Δεδομένων



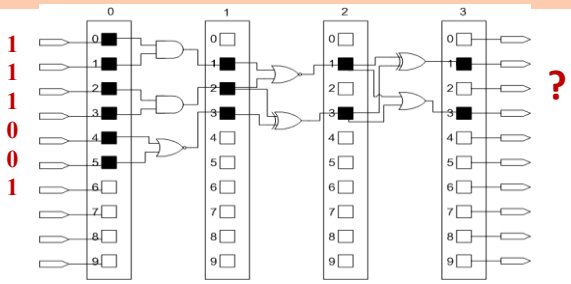
```
#define INPUT_COLUMN  0
#define MIDLE_COLUMN1 1
#define MIDLE_COLUMN2 2
#define OUTPUT_COLUMN 3
```

- **Αναφορά στο breadboard**

```
for(int i=0;i<10;i++)
   breadBoard[INPUT_COLUMN][i]= true;

for(int i=0;i<10;i++)
   printf("%d->%d\n",i,breadBoard[OUTPUT_COLUMN][i]);
```


Ορισμός διεργασιών

```
defineCircuitInputs(){
  // breadBoard[INPUT_COLUMN][0]= ..;
}

calculateOutputs(){
// breadBoard[MIDLE_COLUMN1][0]= ..;
}

printCircuitOutputs(){
//printf("%d->%d\n",i,breadBoard[OUTPUT_COLUMN][i]);
}
```




```
printCircuitStatus(){

}
printCircuit(){

}
```

Το δικό σας κύκλωμα

- Σχεδιάστε ένα δικό σας κύκλωμα λογικών πυλών
- Μπορεί να χρησιμοποιηθεί το πρόγραμμα που αναπτύξατε για να υπολογίσετε την έξοδο του δικού σας λογικού κυκλώματος;
- Τροποποιήστε το κατάλληλα ώστε να σας δώσει την έξοδο για το δικό σας κύκλωμα.

Αναβάθμιση προγράμματος

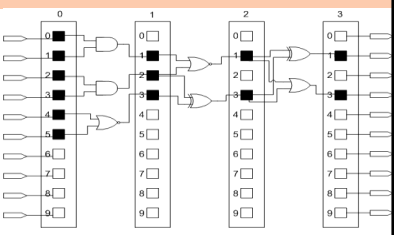
- Αναπτύξτε ένα πρόγραμμα ώστε να προσφέρει στον χρήστη επιπλέον την **υπηρεσία Ορισμός Κυκλώματος Λογικών Πυλών**.
- Ποιες από τις διεργασίες της έκδοσης 1 θα πρέπει να τροποποιήσετε;

Ορισμός Κυκλώματος Λογικών Πυλών

Ερωτήσεις:

Α) Τι πληροφορία πρέπει να δώσουμε στο σύστημα για την περιγραφή (αναπαράσταση) του λογικού κυκλώματος;

Β) Πως θα **οργανώσουμε** την πληροφορία αυτή;



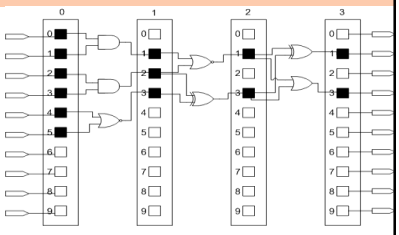
Περιορισμοί (Constraints)

1. Το κύκλωμα μας αποτελείται από τρεις στήλες πυλών.
2. Κάθε στήλη πυλών βρίσκεται ανάμεσα σε δύο στήλες του breadboard
3. Ο μέγιστος αριθμός των πυλών ανά στήλη είναι ορισμένος εκ των προτέρω.

Πληροφορία για περιγραφή κυκλώματος (1/3)

Για κάθε πύλη θέλουμε:

- την τάξη του pin της εισόδου 1 της πύλης
- την τάξη του pin της εισόδου 2 της πύλης
- την τάξη του pin της εξόδου της πύλης
- τον τύπο της πύλης
- την στήλη του board στην οποία βρίσκεται η πύλη



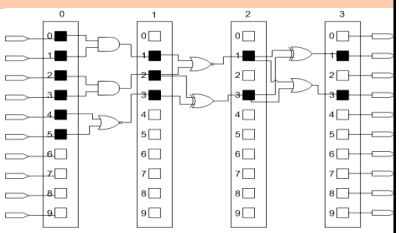
?

Α/Α	Τύπος πύλης	Αριθμός υποδοχής εισόδου 1	Αριθμός υποδοχής εισόδου 2	Αριθμός υποδοχής εξόδου	Αριθμός Στήλης
1					
2					
3					
4					
5					
6					
7					
8					

Πληροφορία για περιγραφή κυκλώματος (2/3)

Για κάθε πύλη θέλουμε:

- την τάξη του pin της εισόδου 1 της πύλης
- την τάξη του pin της εισόδου 2 της πύλης
- την τάξη του pin της εξόδου της πύλης
- τον τύπο της πύλης
- την στήλη του breadboard στην οποία βρίσκεται η πύλη

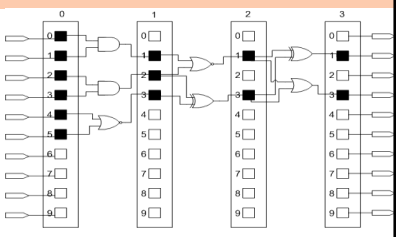


```
int input1PinOrder;
int input2PinOrder;
int outputPinOrder;
int gateType;
```

```
int column1gate1[4];
…
int column1gate8[4];
```

Πληροφορία για περιγραφή κυκλώματος (3/3)

```
int column1gate1[4];
…
int column1gate8[4];
```



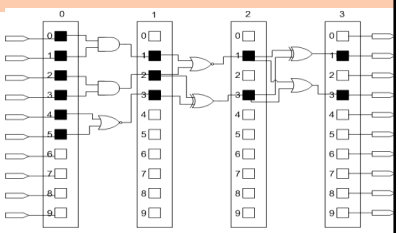
```
int column1Gates[8][4];
int column2Gates[8][4];
int column3Gates[8][4];
```

int circuitGates[3][8][4];

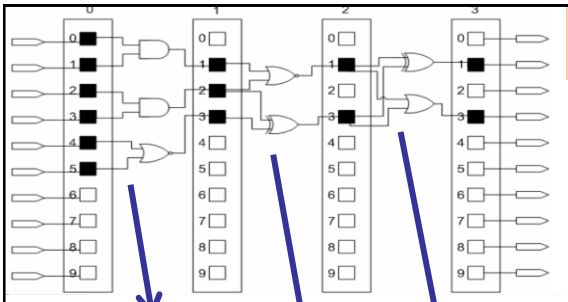
Περιγραφή κυκλώματος – Μορφή κειμένου

AND, C1,0,1,1
AND, C1,2,3,2
NOR, C1,4,3,3
NOR, C2,1,2,1
XOR, C2,2,3,3
…

MyCircuit.txt



Α/Α	Τύπος πύλης	Αριθμός υποδοχής εισόδου 1	Αριθμός υποδοχής εισόδου 2	Αριθμός υποδοχής εξόδου	Αριθμός Στήλης
1					
2					
3					
4					
5					
6					
7					
8					

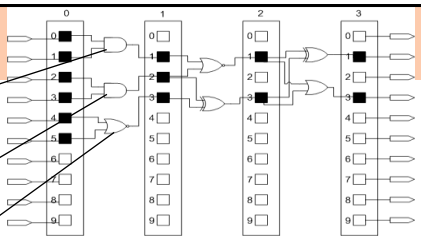


MyCircuit.txt

AND, C1,0,1,1
AND, C1,2,3,2
NOR, C1,4,3,3
NOR, C2,1,2,1
XOR, C2,2,3,3
...

A/A	Τύπος πύλης	Αριθμός υποδοχής εισόδου 1	Αριθμός υποδοχής εισόδου 2	Αριθμός υποδοχής εξόδου
1				
2				
3				
4				
5				
6				
7				
8				

circuitGates[3][8][4]



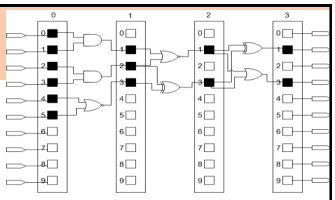
0 (C1),1(C2),2(C3) Στήλη
Τάξη πύλης στην Στήλη

```
gates[0][0][0] = AND; /* είναι η πρώτη πύλη της στήλης και είναι τύπου AND  */
gates[0][0][1] = 0; /* η μία είσοδος θα συνδεθεί στην υποδοχή 0 της στήλης 0 */
gates[0][0][2] = 1; /* η άλλη είσοδος θα συνδεθεί στην υποδοχή 1 της στήλης 0 */
gates[0][0][3] = 1;     /* η έξοδος θα συνδεθεί στην υποδοχή 1 της στήλης 1 */
gates[0][1][0] = AND;
gates[0][1][1] = 2;
gates[0][1][2] = 3;
gates[0][1][3] = 2;
gates[0][2][0] = OR;
gates[0][2][1] = 4;
gates[0][2][2] = 3;
gates[0][2][3] = 3;
```

AND, C1,0,1,1
AND, C1,2,3,2
NOR, C1,4,3,3
NOR, C2,1,2,1
XOR, C2,2,3,3

MyCircuit.txt

Επιλογή Νο 2 – Χρήση Δομής



Πληροφορία για περιγραφή κυκλώματος

Για κάθε πύλη θέλουμε:

την τάξη του pin της εισόδου 1 της πύλης
την τάξη του pin της εισόδου 2 της πύλης
την τάξη του pin της εξόδου 1 της πύλης
τον τύπο της πύλης **(αλφαριθμητικό)**
την στήλη του board στην οποία βρίσκεται η πύλη

```
struct gateType {
    char type[5];
    int input1PinOrder;
    int input2PinOrder;
    int outputPinOrder;
    }
```

```
struct gateType gate1;
```

```
struct gateType circuitGates[3][8];
```


Logic Gates Circuit Sim – OOP - Identify Objects