4. Übungsblatt

10. Aufgabe

Ein Problem bei der Programmierung in C ist, dass zur Laufzeit der Zugriff auf Feldelemente nicht kontrolliert wird, dieser Zugriff auf Feldelemente allerdings nur im gültigen Indexbereich erfolgen darf.

Implementieren Sie in C++ eine **Klasse** Feld, die bei dem Feldzugriff kontrolliert, dass der Index in dem erlaubten Bereich liegt. Damit soll das angegebene Hauptprogramm ablaufen.

```
main ( )
{
 int i;
 Feld v1(1,3), v2(-5,+5), v(10);

 if (v1.size() == v2.size())
   printf ("v1 und v2 haben die gleiche Dimension");

 i = v2[-1];
 v.display();
}
```

Die Klasse Feld soll folgende Eigenschaft besitzen:

- Die Inhalte des Feldes sind vom Typ integer.
- Es soll möglich sein eine obere und eine untere Indexgrenze angeben zu können. Die möglichen Indices sind dann **untere Indexgrenze bis obere Indexgrenze -1**.
- Bei der Instanziierung von Variablen werden alle Werte des Feldes auf 0 initialisiert; das Feld wird dynamisch erzeugt. Falls bei der Instanziierung oben nicht größer oder mindestens gleich unten ist, sollen oben und unten vertauscht werden.
- Wird versucht, auf ein nicht vorhandenes Feldelement zuzugreifen, dann erfolgt ein kurzer Hinweis und der Wert –1 wird zurückgegeben
- Wenn der Konstruktor nur einen Wert erhält, dann soll dieser oben entsprechen und unten den Wert 0 haben.
- Die Größe eines Feldes soll per size() abgefragt werden können.

11. Aufgabe (**Praktikum**)

Neuronale Netze können unter anderem als Erkennungskomponente verwendet werden. Mögliche Anwendungen sind Beispielsweise:

- Mustererkennung
- Zeichenerkennung
- Gestenerkennung
- … …

Realisieren Sie ein Programm, das mit Hilfe eines objekt-orienterten neuronalen Netzes die Identifikation eines „verrauschten" Musters durchführt. Hierzu soll Ihr neuronales Netz zuerst 4 verschiedene Orginal-Muster erlernen. Nach der Lernphase soll Ihr neuronales Netz dann zu einem gegebenen verrauschten Muster das zugehörige Original ausgeben.

Implementieren Sie dieses Programm unter Nutzung folgender Zusammenhänge:

- Die Konversion eines binären Wertes (0,1) in einen bipolaren Wert (-1,1) erfolgt gemäß:
  $x_{binär} = (2*x - 1)_{bipolar}$
- Während der Lernphase berechnen sich die Neuronengewichte gemäß:

  $w_{ji} = \sum x_{sj} * x_{si}$ wenn $i \neq j$

  $w_{ji} = 0$ wenn $i = j$

  wobei s = 1 .. p (= Anzahl Pattern) ist
- Während der Identifikationsphase berechnet sich die Neuronenausgänge gemäß:

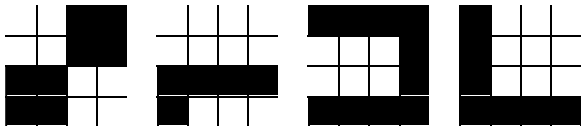
  $Net_j = sign (\sum w_{ji} * x_i^{bipolar})$ mit i = 1 .. m (= Anzahl Pixel)
- Das Ergebnis eines Identifikationsschrittes bildet solange die Eingabe für den nächsten Identifikationsschritt bis sich die Ausgabe nicht mehr von der Eingabe unterscheidet. Unterscheiden sich die Eingabe und die Ausgabe nicht mehr, wurde das Muster erfolgreich identifiziert.

Entwerfen und Implementieren Sie zuerst Klassen für:

a) Die Muster
b) Ein Neuron
c) Das Neuronale Netz

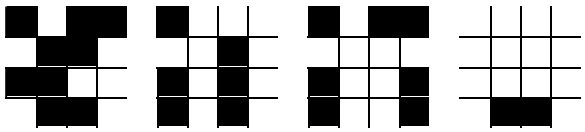
Schreiben Sie dann Ihr Hauptprogramm mit einem 16-Neuronen-Netz das folgende Ausgaben erzeugen/protokolliert (0 ≡ weiß, 1 ≡ schwarz) kann:
Zu trainierende Muster für das Neuronale Netz:

Orginale



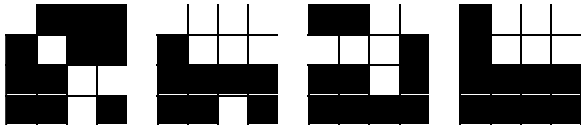
Nach der Trainingsphase sollte Ihr Neuronales Netz folgende verrauschte Muster gemäß der angegebenen Iterationen erkennen.

Eingabe

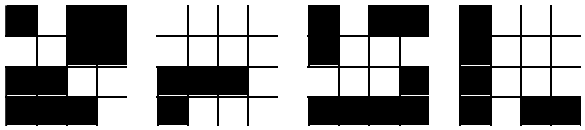


Iteration

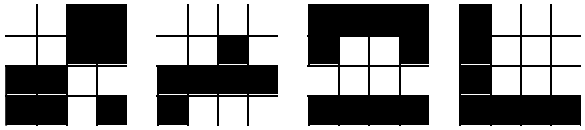
1



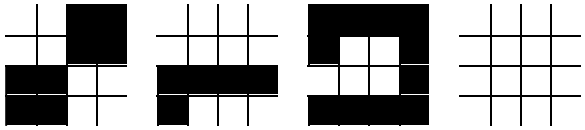
2



3



4



5