מת"מ - 234124

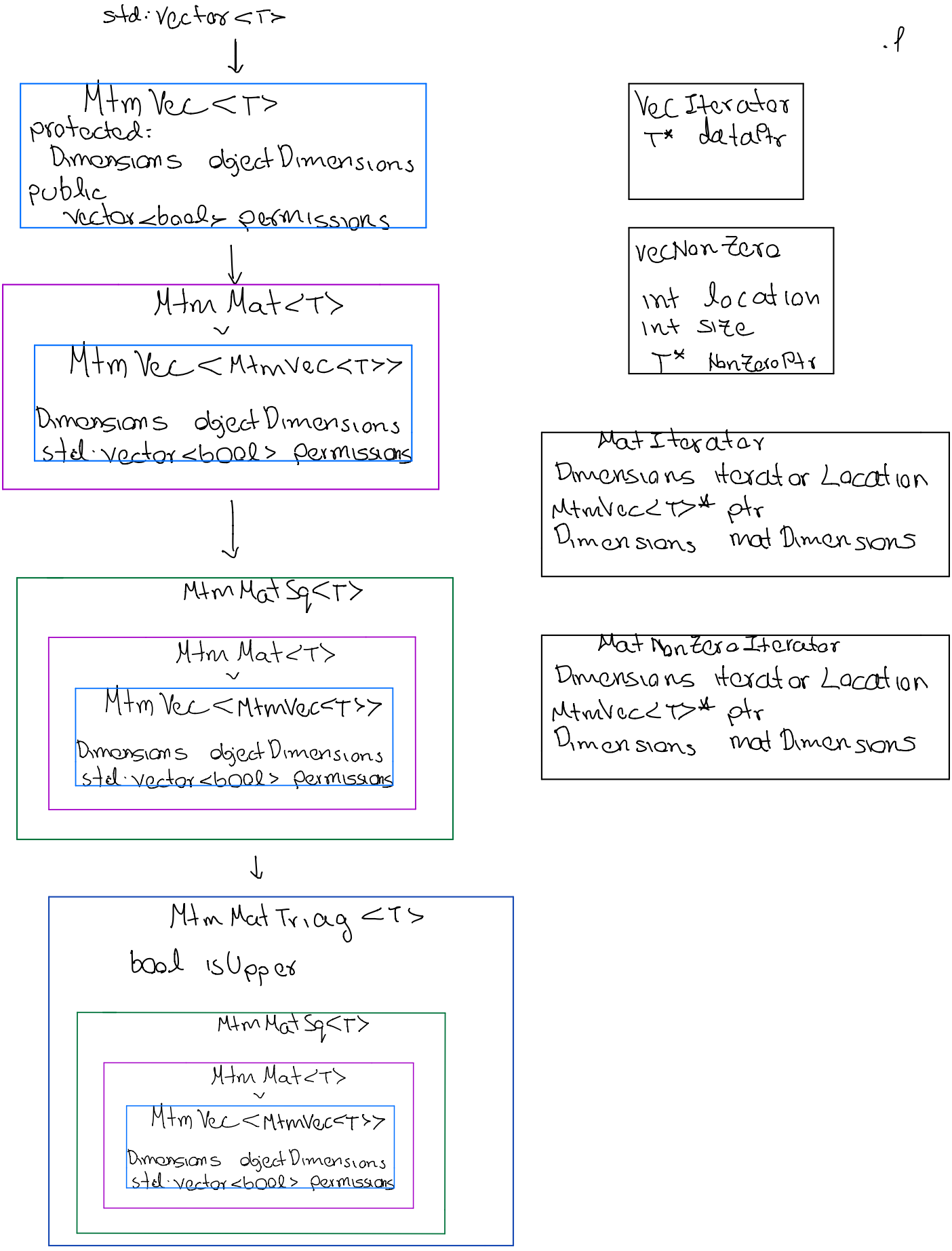
תרגיל בית 3 – יבש

מגישים:

בר רוה
bar.raveh@campus.technion.ac.il
208950238

קשת מאיר
keshstmeir@campus.technion.ac.il
312179674

תאריך הגשה: 2019-1-24


std::vector<T>
MtmVec<T>
protected:
Dimensions objectDimensions
public
vector<bool> permissions
MtmMat<T>
MtmVec<MtmVec<T>>
Dimensions objectDimensions
std::vector<bool> permissions
MtmMatSq<T>
MtmMat<T>
MtmVec<MtmVec<T>>
Dimensions objectDimensions
std::vector<bool> permissions
MtmMatTriag<T>
bool isUpper
MtmMatSq<T>
MtmMat<T>
MtmVec<MtmVec<T>>
Dimensions objectDimensions
std::vector<bool> permissions
VecIterator
T* dataPtr
vecNonZero
int location
int size
T* NonZeroPtr
MatIterator
Dimensions iteratorLocation
MtmVec<T>* ptr
Dimensions matDimensions
MatNonZeroIterator
Dimensions iteratorLocation
MtmVec<T>* ptr
Dimensions matDimensions

1. כל מה שלא כתוב עליו במפורש הוא private.
בחרנו בירושה זו כיוון שמטריצה אלכסונית היא סוג של מטריצה מרובעת, מטריצה מרובעת היא סוג של מטריצה רגילה, ומטריצה היא וקטור של וקטורים. objectDimensions מוגן כי היה צורך לגשת אליו גם במחלקות היורשות, וpermissions הוא פומבי כי רצינו להיות מסוגלים לערוך אותו מחוץ לאובייקט.
isUpper הוא פרטי כי שום דבר לא יורש ממטריצה אלכסונית לכן אין צורך להשתמש בו מבחוץ.
האיטרטורים מוגדרים באופן בלתי תלוי זה בזה, לכן לא הייתה סיבה להגדיר את המשתנים שלהם כלא פרטיים.
2. מטריצות אלכסוניות הן סוג של מטריצות משולשות, לכן המימוש האופטימלי יהיה כמחלקה יורשת של מטריצה משולשת.
3. לשמור מטריצה דלילה כיורשת של מטריצה רגילה או יהיה בזבוז גדול של זכרון, כי רוב המקום בזיכרון ילך על אפסים.
פתרון אפשרי וחסכוני יותר בזיכרון יהיה לממש מטריצה דלילה כיורשת של וקטור, כאשר כל איברי המטריצה שאינם אפס נמצאים בוקטור עמודה אחד, ולייצר וקטור int תואם המכיל את האינדקס הליניארי של כל איבר שאינו אפס.
פתרון נוסף אשר כן עושה שימוש במטריצה יהיה לרשת ממטריצה ולהשתמש בnziterator בעת שימוש באופרטורים.
4. הבעיה בבקשה זו היא שהיא תדרוש מאיתנו להגדיר מספר גדול מאוד של המרות בין סוגי טיפוסים אפשריים (המרה ממטריצת int למטריצת Complex וכו'). טעויות בהמרות שכאלו יכולות לגרור המרות אוטומטיות לא צפויות ובאגים.
5. זהו מנהג תכנותי נכון כיוון שככה נמנעות התנגשויות בין שמות של פונקציות. בנוסף בצורה זו אפשר להשתמש בנוחות בכל הפונקציות תחת הnamespace.
6. מכמה סיבות – העברת רפרנס לאובייקט לפונקציה ולא את האובייקט עצמו חוסכת מקום במחסנית הקריאות, העברת אובייקט by reference מאפשרת לנו לשנות את תוכן הרפרנס לעומת העברה by value שלא הייתה מאפשרת לערוך את האובייקט שקיבלנו, וזו דרך "בטוחה" יותר לעבוד במקום פוינטרים בC.