שפות תכנות, 236319

חורף 2020-2021

תרגיל בית 5

תאריך פרסום: 29.12.2020
מועד אחרון להגשה: 12.1.2021
מועד אחרון להגשה מאוחרת: 16.1.2021
מתרגל אחראי: מרח גומייד
אי-מייל: marah@campus.technion.ac.il
בפניה בדוא"ל, נושא ההודעה (subject) יהיה "PL-EX5" (ללא המירכאות).

תרגיל בית זה מורכב משני חלקים, חלק יבש וחלק רטוב.
לפני ההגשה, **ודאו שההגשה שלכם תואמת את הנחיות ההגשה בסוף התרגיל.**
תיקונים והבהרות יפורסמו בסוף מסמך זה, אנא הקפידו להתעדכן לעתים תכופות

חלק יבש

שאלה 1- RTTI

1. קראו את פרק 5.6 בחוברת השקפים בנושא **RTTI**.
מהו מנגנון RTTI? סכמו בשני משפטים.

מנגון ה-Run-time type information או בקיצור RTTI , הוא מנגנון המצמיד סימון לכל ערך שמגדיר את הטיפוס שלו.

2. האם בשפת C יש מנגנון RTTI?
אם כן - תארו אותו.
אם לא - הסבירו למה.

לשפת c אין מנגנון כזה מתכונת ה-"no hidden cost" בחזון של שפת c השפה לא מסתירה מהמשתמש פעולות יקרות למימוש כפי שקורה לדוגמה בjava עם מנגנון ניהול הזיכרון.

3. הסבירו: כיצד מנגנון איסוף אשפה משתמש במנגנון RTTI?

הgarbage collector ממפה את הזיכרון על מנת לדעת איזה חלקים בו עדיים רלוונטים ואיזה לא. כמו כן, חלק ממנגנוני איסוף האשפה מעתיקים חלקים מהזיכרון. לצורך כך, כאשר המנגנון מגיע לערך שהוא עדיין בשימוש, עליו לפענח כמה מקום הוא תופס בזיכרון וכיצד לפענח אותו -אם הביטוי הוא מורכב ולא מטיפוס בסיסי (כך שאם למשל הערך מכיל רפרנס לערך אחר המנגנון ידע לסמן גם ערך זה כמשומש, ולצורך כך המנגנון צריך להשתמש בRTTI).

4. תארו שימוש אחר (שאינו איסוף אשפה) למנגנון RTT בשפות הדוגלות ב static typing

RTTI יכול להועיל גם בשפות שבהן יש מערכת טיפוסים סטטית במקרה שנרצה לעשות העתקה עמוקה של אובייקט.
לצורך העתקה "עמוקה" של ערך, יש לעבור על כל הערכים שהאובייקט מכיל (מצביע עלהים) ממנו, ולהעתיק גם אותם. יש לסרוק את רשת המצביעים המתחילה בערך המועתק ומכילה את כל הערכים שאליהים ניתן להגיע ממנו.
לצורך סריקה זו יש להתחיל מהערך הנתון, לסמן שביקרנו בו, ולחזור על התהליך עם כל אחד מהערכים המוצבעים ממנו. אולם, כשנגיע לערך מסוים לא נדע מה הוא, מהם הרכיבים שלו ומהם הערכים הישיגים(שניתן להגיע אליהם) ממנו. ללא מידע נוסף, ערך שמגיעים אליו בסריקה הוא זיכרון שלא ידוע מה גודלו ומה רכיביו - לא ידוע הטיפוס של הערך. על מנת "לפרש" את הזיכרון, על אלגוריתם הסריקה לדעת מהו הטיפוס של הערך ומהם הטיפוסים של הערכים המוצבעים ממנו. לצורך ידיעת הטיפוס בזמן ריצה יש צורך ב-RTTI, השומר מידע על הטיפוס לצד הערך בזמן ריצה.

שאלה 2- מערכים וזיכרון

1. ציינו את הדומה והשונה בין static array ו dynamic array.

מערך סטסטי:
גודלו נקבע בזמן הקומפילציה , מוקצה ב-data segment , גודל קבוע, קבוצת האינדקסים נקבעת בזמן הקומפילציה.
מערך דינאמי:
גודלו נקבע בזמן הריצה, מוקצה ב-heap segment , גודל קבוע, קבוצת האינדקסים נקבעת בעת יצירת משתנה המערך.
דומה: לשניהם יש מימוש יעיל במודל הזיכרון הקלאסי לעומת מערכים אחרים. גודלם קבוע.
שונה: זמן קביעת הגודל, המקום שבו המערכים מוקצים, זמן קביעת קבוצת האינדקסים.

2. ציינו את הדומה והשונה בין stack based array ו associative array.

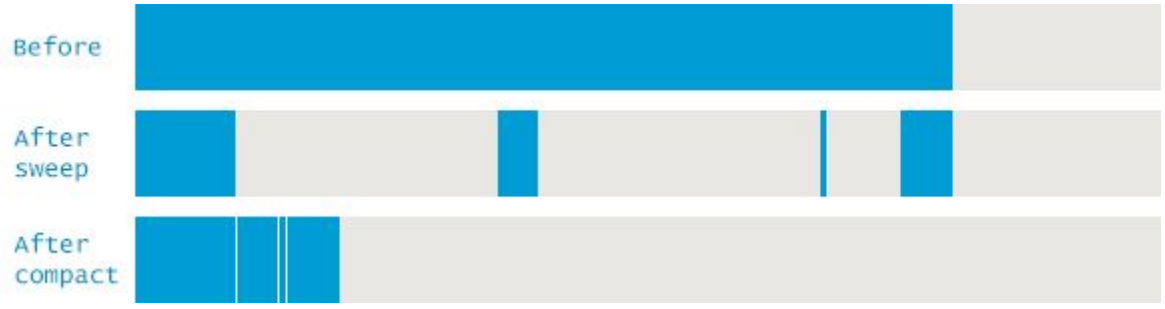
מערך STACK BASED - גודלו נקבע בזמן הריצה, ואינו יכול להשתנות לאחר מכן. מוקצה על המחסנית. קבוצת האינדקסים נקבעת בזמן יצירת המערך.

מערך ASSOCIATIVE - האינדקסים יכולים להיות מכל טיפוס, טיפוס אופייני לשם כך הוא מחרוזות. מימוש אופייני הוא באמצעות HASH TABLE. אין הגבלה על קבוצת האינדקסים, הקבוצה משתנה כשערכים מתווספים למערך או מוסרים ממנו.

שונה: מבוסס מחסנית קבוצת האינדקסים נקבעת בזמן יצירת המערך לעומת מערך אסוציטיבי שבו היא לא מוגדרת ומשתנה כאשר מוסיפים איבר למערך או מוציאים איבר. אופן המימוש יהיה שונה: אסוציאטיבי באמצעות טבלת ערבול ואילו מערך מבוסס מחסנית ישתמש בהזזת הstack pointer. קבוצת האינדקסים של מערך אסוציטיבי יכולה להיות מכל טיפוס.

דומה:לשניהם נקבע הגודל בזמן הריצה.

3. בחרו אלגוריתם ניהול זיכרון אוטומטי שלא נראה בכיתה. הסבירו את האלגוריתם לשלביו, ציינו את היתרונות שלו, את החסרונות שלו (ביחס לאלגוריתמים האחרים שהוצגו, וגם ביחס לאפשרות של ניהול זיכרון ידני).



האלגוריתם שבחרנו מתנהג כמו אלגוריתם mark and sweep באופן כזה שהוא מבצע את אותם השלבים של mark and sweep (כלומר עובר אובייקט אובייקט ומסמן את מי מהאובייקטים יש למחוק (before בתמונה) לאחר מכן עובר ומוחק את כל האובייקטים שסומנו כדבל ומוחק אותם (after sweep בתמונה)(אם

התייחסנו לזוגיות של התהליך בעת הסימון אז אין צורך לעבור ולהוריד את הסימון מכל האובייקטים שבחרנו להשאיר אם לא עלינו לעשות זאת...) בשלב זה היינו מסיימים את הריצה של אלגוריתם mark and sweep והיינו מקבלים זיכרון פנוי שאינו רציף (after sweep בתמונה). ב-MARK SWEEP and COMPACT אנחנו נבצע את אותם השלבים ולאחר מכן נעבור על כל התאים שמכילים אובייקטים ונעתיק אותם אל תחילת ה heap ככה נקבל את כל הזיכרון שפנוי כזיכרון רציף (after compact בתמונה) כשנסיים נחזיר מצביע אל המקום בו נמצא סוף האובייקט האחרון כלומר האובייקט הימני ביותר. כאשר נרצה להקצות אובייקט חדש יש בידינו אפשרות להקצות מקום לאובייקט בגודל כל הזיכרון שהיה פנוי (מאחר וכעת הוא רציף). ה-heap משמאל למצביע מלא ומימין פנוי להקצאה.

עובדה זו שהוספנו שלב לתהליך תעלה לנו בזמן הריצה שכן אנחנו מבצעים העתקה של (ייתכן) כל האובייקטים ברשותנו, דבר זה גוזל זמן ומעכב את קבלת המקום הפנוי להקצאה.

פרמטר השוואה /אלגוריתם	Mark and Sweep	Stop and Copy	MARK SWEEP and COMPACT Algorithm
יכולת להקצות אובייקט גדול	ניתן להקצות אובייקט בגודל הזיכרון הרציף הגדול ביותר.	מגביל עד לגודל התא הגדול מבין השניים (האזור החם או הקר).	מאפשר להקצות אובייקט בגודל כל המקום הפנוי.
זמן המתנה על מנת לקבל מקום פנוי להקצאה	יש להמתין שהאלגוריתם יסתיים על מנת להקצות שיהיה אפשר להקצות מקום	מבצע במקביל את הפנוי של האיזור (החם שהפך לקר) וניתן להקצות מקום לאחר שמתבצעת ההעתקה של כל האוביקטים (מהזור שהיה חם לאזור החם העכשווי) אין צורך להמתין לפנוי של האזור (הקר העכשווי) .	ש להמתין שהאלגוריתם יסתיים על מנת להקצות שיהיה אפשר להקצות מקום וייקח זמן רב יותר לסיומו מאחר וייתכן ומתבצעת העתקה של כל האובייקטים הקימיים לתחילת המחסנית.ד
מתי מתבצע עדכון	ניתן להחלטה של המערכת אם כל כמות מסויימת של בתים שהוקצאה או כל כמות זמן מסויימת.	כאשר האזור החם מתמלא מתבצעת העתקה לאזור הקר רק עם הפרמטרים הרלוונטים דבר זה מעניק יתרון בכך שהוא מבצע במקביל את הפנוי ואין צורך לחכות שסיים את הפנוי של האיזור (החם שהפך לקר).	באופן דומה Mark and Sweep רק שתהיה המתנה נוספת בכל ריצה.

שאלה 3- אוסף שאלות קצרות

1. מה ההבדל בין type error ל pseudo type error, תנו דוגמה לכל אחד.

שגיאת טיפוס type error מתרחשת כאשר אנו מנסים לבצע פעולה כלשהי על ערך v מטיפוס T באופן שאינו מוגדר על הטיפוס T.

בשונה מזה, שגיאת pseudo type error זוהי שגיאת הנוצרת גם מהפעלת פעולה על ערך v מטיפוס T,

רק שבמקרה כזה, הפעולה אכן מוגדרת על הטיפוס T, אך באופן ספציפי אינה עובדת עבור v.

```c
#include <stdio.h>

int main()
{
    int x =1, y=2;
    int* xp=&x;
    int * yp = &y;
    int sub;
    sub = xp*yp;
}
```

```c
#include <stdio.h>

int main()
{
    int x = 1, y = 0;
    int d = x/y;
}
```

הדוגמה השמאלית מתארת מצב בו פעולת הכפל אינה מוגדרת על פוינטרים לטיפוסים מסוג int, ולכן ישנה שגיאת טיפוס.

הדוגמה הימנית ישנו תיאור מצב של חילוק בין שני int שנכשלת מכיוון ש חילוק באפס אינו מוגדר, למרות ש באופן כללי הפעולה חוקית עבור טיפוסים אלו.

2. ציינו את כל שיטות שערוך שראינו בקורס, כתבו הסבר קצר על כל אחת ותנו דוגמת קוד קצרה בסינטקס דמוי C, שעבורה היה פלט שונה עבור כל שיטת שערוך שראינו.

שערוך רקורסיבי

שערוך קפדני-egar שערוך זה משומש על ידי רוב השפות כיום, בשערוך זה בכל השמה של ביטוי למשתנה משערכים את הביטוי ברגע ההשמה.

שערוך קפדני- EVALUATION STRICT הנקרא גם EVALUATION EAGER או APPLICATIVE , כל הפרמטרים מחושבים לפני חישוב הביטוי / הפעלת הפונקציה. גם

אם אולי לא יהיה בהם צורך, כל חלקי הביטוי מחושבים מראש.EVALUATION CIRCUIT SHORT -ערכים לא מחושבים אם אין בהם צורך. למשל בחישוב תוצאות של פעולות בוליאניות or, and.

שערוך נורמלי - פרמטרים לפונקציה מחושבים רק כאשר יש בהם שימוש בפונקציה. זהו מקרה פרטי של SHORT CIRCUIT EVALUATION.

שערוך עצל -EVALUATION LAZY הביטוי משוערך בפעם הראשונה בלבד שבה הפונקציה הנקראת משתמשת בפרמטר.

```
#include <stdio.h>
int m = 0;
int n = 0;
int f(int x){
 m++;
 return 1;}

int g(int x){
return x + x;}

int h(int x){
n++;
return x;}

void main(){
int x = g(f(h(1)));
        if(m==2){ //normal, f will be evaluated 2 times.
                printf("normal"); }
        else if(m==1){
                if(n==1){//eager, f will be evaluated one time and h is evaluated
once.
                        printf("eager");}
                else if(n==0){//lazy, f will be evaluated one
                        time and there is no need to evaluate h.
                        printf("lazy");}
         }
        }
```

3. הסבירו את משמעות הביטוי- "בנאי אורתוגונלי".

בנאי אורתונולוגי הוא בנאי שאינו מפלה כלומר הוא מפעיל את פעולתו על כל טיפוס בשפה, דוגמא לבנאי לא אורתונולוגי הוא פונקציות בשפת C שלא מקבלות מערך כקלט או פלט ולכן מפלים אותו.