הנחיות כתיבת קוד(Coding Style)

הקדמה

בניגוד לדעות הנפוצות רוב זמנם של מתכנתים אינו מוקדש לכתיבת תכניות. רוב זמנו של המתכנת מוקדש לתחזוקה , שדרוג ותיקון תקלות בקוד קיים. רוב הקוד בימינו בנוי על קטעי קוד קיימים וכל שפת תכנות מגיעה עם ספריית פונקציות מגוונת המוכנה לשימוש. קוד שאינו כתוב בצורה פשוטה ואינו מכיל הערות והסברים בתוכו הינו קוד שיהיה מאוד קשה ומסובך לתקן, לשדרג או לתחזק בעתיד .

1. רקע

1.1. מטרת מסמך זה הוא הצגת סגנון ואופן כתיבת קוד ANSI-C ו python כפי שנדרש בקורס. כל קובץ קוד שיוגש במהלך השנה עלול להיבדק בשני אופנים: נכונותו (בעזרת המערכת האוטומטית) וסגנון הכתיבה (בעזרת בודק ידני).

1.2. באופן כללי, במהלך כתיבת קוד אתם נדרשים לשמור על העקרונות הבאים :

- קוד צריך להיות נטול שגיאות וחסין לטעויות משתמש .
- קוד צריך להיות קל לשימוש ולהבנה .
- קוד צריך להיות פשוט לתחזוקה .

ANSI-C

2. הידור הקוד

2.1. הקוד צריך להתקמפל (לעבור הידור) בקומפיילר gcc ללא אזהרות (warnings) .
במידה ואין אפשרות לנקות אזהרה מסוימת, חובה לתעד את סיבת הופעתה בקוד המקור .

3. כללי שמות

3.1. שם מזהה אמור לספק מידע ולתאר את בעליו .

3.2. אסור לתאר שם משתנה בעזרת המילה העברית שלו, לדוגמא govaHesbun .

3.3. שמות משתנים : שם משתנה יהיה קצר ובעל משמעות ויתחיל באות קטנה כאשר כל מילה נוספת המורכבת לשם תתחיל באות גדולה, לדוגמא: playerHeight .

3.4. שמות פונקציות : שם פונקציה אמור לתאר במידת האפשר את הפעולה המבוצעת על ידי הפונקציה. הוא יתחיל באות גדולה וכל מילה נוספת המורכבת לשם תתחיל באות גדולה, לדוגמא: SetTime .

3.5. שמות טיפוסים נוספים: קבועים של define # יהיו כולם באותיות גדולות וכל מילה תופרד על ידי קו תחתון , לדוגא DAYS_IN_WEEK.

3.6. שמות enum: שם הטיפוס יהיה באותיות גדולות והערכים באותיות קטנות. לדוגמא:

```
enum DAY {SUNDAY,MONDAY};
enum DAY today;
```

3.7. שמות קבצים:

- שם קובץ יתאר את תוכנו או תפקידו של הקובץ. אלא אם יצוין מפורשות בתרגיל מה שם הקובץ שיש להגיש .
- קבצי header יקבלו את הסיומת ".h"
- קבצי מקור (source) יקבלו את הסיומת ".c"

- שם קובץ יכול להכיל את תווי האותיות, מספרים וקו תחתון.

4. משתנים, אופרטורים וביטויים

4.1. יש להמנע משימוש במשתנים גלובלים ככל האפשר. במידה ומשתמשים בהם, יש לתעד בקוד את הסיבות לכך

4.2. יש לתת לקומפיילר לחשב את גודלם של מבנים: השתמשו באופרטור sizeof ולא בכתיבה מפורשת של גודל המבנה .

4.3. עדיף להשתמש ב cast-ישירות בזמן המרת משתנים ולא לסמוך על ההמרה האוטומטית של המהדר .

4.4. השמות של מצביעים צריכות להעשות רק כאשר הם מאותו סוג .

4.5. על מנת לבדוק חוקיות של מצביע, יש להשוותו לקבוע NULL ולא לערך 0 .

5. משפטים וזרימה

5.1. תמיד כיתבו תנאי default למשפט switch.

5.2. בלולאות for לעולם לא ישונה ערך מונה הלולאה בתוך הלולאה עצמה .

5.3. בלולאות while , שינוי ערך המונה יעשה בתחילת הבלוק או בסופו בלבד .

6. הערות קוד

6.1. הקבצים המוגשים (קבצי header וקבצי source בלבד) יכילו בתחילתם בלוק הערות לקובץ לפי הפורמט הבא :

```
/*****************************************

*Student name: שם הסטודנט פרטי ושם משפחה

*Student ID: ת"ז של הסטודנט

*Course Exercise Group: קבוצת ההגשה – מדמ"ח 01\מתמטיקה 02

*Exercise name: שם התרגיל

*****************************************/
```

למשל:

```
/*****************************************

* Student name: Avi Cohen

* Student: 123123123

* Course Exercise Group: 01

* Exercise name: Exercise 1

*****************************************/
```

6.2. מטרתם של הערות קוד הינם לתת תיאור מילולי (באנגלית) של קטעי קוד, תיאור אלגוריתם והסברים שונים על קטעי קוד שאינם ברורים במבט מהיר. כמות ההערות קוד אינה מדע מדויק, אבל חוק אצבע ראשוני שייתן לכם את הכיוון הכללי לכמות ההערות הינו שורת הערה אחת לשלוש שורות קוד.

6.3. מותר להשתמש בהערות בלוק ובהערות שורה (בעזרת //) לפי הכללים הבאים:

- הערות בלוק – חובה להכניס שורת רווח לפני הכנסת הערות בלוק, והזחות ההערה צריכות להיות לפי הזחות הקוד שההערה מתייחסת אליו ולפי הפורמט המתואר בדוגמא שלפניכם .
- הערות שורה – הינם הערות קצרות לתיאור משתנים וקטעי קוד פשוטים. הם חייבים להגיע לפני השורה שאליה מתייחסת ההערה (ולא בהמשך השורה או מתחתיה) והזחות ההערה צריכות להיות לפי הזחות הקוד שאליה היא מתייחסת. אסור לשבור הערות שורה ליותר משורה אחת .

```
while (i > 0){

        // line comment
        int counter;

        /*

        block comments……

        */

        if (…..){

                ….

        }

}
```

6.4. להלן פורמט ההערות לפיו אתם נדרשים להשתמש לפני כל פונקציה (מספר הכוכביות לא חשוב):

```
/*********************************************************************
* function name: שם הפונקציה                                          *
* The Input: הסבר על הקלט לפונקציה                                   *
* The output: הסבר על הפלט של הפונקציה                               *
* The Function operation: דרך הביצוע הכללית                           *

*********************************************************************/
```

7. קריאות הקוד

7.1. כתוב פקודת קוד אחת בלבד כל שורה (אין להכניס שתי פקודות בשורה אחד מופרדות ב"; ")

7.2. בביטויים מורכבים ארוכים מעל 20 שורות יש לשים הערת שורה מיד לאחר סוגר הבלוק, לפי הפורמט הבא:

```
{

 …;

 .

 .

 .

 …;

} // end of while loop
```

7.3. הזחות הקוד צריכות להיות לא גדולות מ 4 רווחים, במידה והקוד מכיל שורות ארוכות ניתן להשתמש ב 3 רווחים, ובשני המקרים חובה לשמור על אחידות הקוד.

7.4. אורך שורה לא יחרוג מ-80 תווים לרוחב .

7.5. במידה וצריך לחתוך משפט לשתי שורות, השורה השניה תתחיל בתחילת הביטוי שנחתך, לפי הדוגמאות הבאות :

```
a = (b + c) *
     (c + d)
-------------------------------------------

 if ((a == b) &&
    (c == d))
-------------------------------------------

 printf("%d %s,
         (a + b),
         ("abcd") )
```

7.6. בפתיחת בלוקים או פונקציות ניתן לבחור באחת משתי צורות הכתיבה הבאות (וחובה לשמור על אחידות צורת הכתיבה לאורך כל התוכנית והקבצים המוגשים):

```
void GetSum(int x) {                    ….;

 ….;                                    }
```

```
if (…){
 ….;
 ….;
}
```

או

```
void getSum(int x)
{
 ….;
 ….;
}
```

```
if (…)
{
 ….;
 ….;
}
```

8. הערות

8.1. הימנעו במידת האפשר מעודף שימוש בקוד מכונן (לדוגמא: ביטוי if בתוך ביטוי if וחוזר חלילה), במידה וההזחות מתקרבות לשוליים הימניים, זה הזמן לבדוק את הקוד ולחלק אותו על מנת לפשט את הקריאה .

8.2. קוד צריך להיות נכון, קריא ופשוט להבנה. לעיתים שני התכונות האחרונות ברשימה הם החשובות ביותר . לפניכם דוגמא לקוד דומה בגרסה קלה להבנה ובגרסה מסובכת להבנה :

```
(// poor programming )
temp = box_x1;
box_x1 = box_x2;
box_x2 = temp;
temp = box_y1;
box_y1 = box_y2;
box_y2 = temp;
----------------------------------------------------------------
/*
 * Swap the two corners
 */
```

```
// Swap X coordinate

temp = box_x1;

box_x1 = box_x2;

box_x2 = temp;

// Swap Y coordinate

temp = box_y1;

box_y1 = box_y2;

box_y2 = temp;
```