

Coding Style

מבוא למדמ"ח תשפ"ה

14 בדצמבר 2024

מסמך זה אינו המלצה, וחירגה מהכללים הכתובים במסמך זה תגרור חורדת ניקוד בתרגילי הבית.
לכן, עליים לעבוד על פי *coding style* שבמסמך זה.

1. הקדמה

בניגוד לדעות הנפוצות, רוב זמנים של מתכנתים אינם מוקדש לכיתבת קוד חדש.
רוב זמנו של המתכנת מוקדש לתחזוקה, שדרוג ותיקון תקלות בקוד קיימים.
רוב הקוד בימינו בניו על קטיעי קוד קיימים וכל שפת תכנות מגיעה עם ספריות וfonkcioot מגוונות המוכנות לשימוש.
קוד שאינו כתוב בצורה פשוטה ואינו מכיל הערות והסברים בתוכו הינו קוד שיהיה מאוד קשה ומסובך לתקן, לשדרוג או לתחזק בעתיד.
מטרת מסמך זה היא הצגת סגנון ואופן כתיבת קוד *C* כפי שנדרש בקורס.
באופן כללי, במהלך כתיבת קוד אטם נדרש לשמור על העקרונות הבאים:

- קוד צריך להיות נטול שגיאות וחסין לטעויות משתמש.

- קוד צריך להיות קל לשימוש ולהבנה.

- קוד צריך להיות פשוט לתחזוקה.

מסמך זה מכיל דרישות נפוצות ונחשב מינימלי בהיקפו.

מי שמעוניין לאמץ הרוגלי סגנון כבר בתחילת הדרך, מוזמן לעיין בקישורים הבאים:

1. [קישור ראשון](#)

2. [קישור שני](#)

3. [קישור שלישי](#)

אך אין חובה לעיין בהם.

1.1. הידור הקוד

הקוד צריך להתකמל על שרטתי האוניברסיטה ללא שגיאות (*warnings*) או אזהרות (*errors*).
בכל תרגיל - פקודת הקימפול תהיה כתובה בתרגיל עצמו.
על כל פנים, חובה לקבל 100 במשוב האוטומטי. (מה שלא מבטיח 100 בבדיקה הסופית).

1.2. הגדרות לשימוש בהמשך

PascalCase 1.2.1

כתיבה באותיות קטנות ללא רווחים, אך כל מילה מתחילה באות גדולה.

camelCase 1.2.2

כתיבה באותיות קטנות ללא רווחים, אך כל מילה נוספת (מלבד הראשונה) - מתחילה באות גדולה.
אם מדובר במילה יחידה? אז רק אותיות קטנות.

UPPER_CASE 1.2.3

כתיבה באותיות גדולות בלבד, הפרדה בין מילים באמצעות קו תחתון.

snake_case 1.2.4

כתיבה באותיות קטנות בלבד, עם קו תחתון מפheid בין מילה למילה.
זהו הסגנון הפופולרי בפייתון.

2 שמות

2.0.1 שמות עבריים בתעתיק אנגלי

אין להשתמש בשמות עבריים בתעתיק אנגלי בתוך הקוד.
לדוגמה - `.aniTov,shemPrati`

2.0.2 שם משתנה

שם משתנה יהיה קצר ובעל משמעות ויכתב בסגנון `.camelCase`
לדוגמה - `.firstName,playerHealth`
משתנה שהשימוש בו קצר טוויך יכול לקבל שם קצר וסתמי (x, y, i, j, k)

לדוגמה :

```
for (i; i < playerHealth; i++)
{
    score += 100;
}
```

ניתן לחסיט בקלות לנחש מה הולאה עשויה.
על כל חיים שיש לשחקן, הניקוד שלו עולה ב100.
זהו משתנה קצר טוויך (הולאה בלבד), ואילו החיים של השחקן - משתנה ארוך טוויך ולכן בעל משמעות.

2.0.3 שם פונקציה

שם פונקציה צריך לתאר במידת האפשר את הפעולה המבוצעת על ידי הפונקציה ויכתב בסגנון `.camelCase`
לדוגמה - `.getInput, setTime`

2.0.4 שמות פונקציות בוליאניות ומשתנים בוליאניים

שמות אלו יתחלו במילה `is` (`isValid, isEmpty`), כאילו מתחילה שאלת.
במקרים מסוימים, ניתן למילה אחרת תואם יותר - אבל עדין יש לשמור על הרעיון של שאלה.
לדוגמה - `.shouldSort, canEvaluate`

2.0.5 קבועי `#define`

הנ'ל יהיו בסגנון `UPPER_CASE`
לדוגמה - `.MAX_INPUT, PI`

enum 2.0.6

שם הטיפוס יהיה בסגנון `camelCase, UPPER_CASE, PascalCase`, הערכים יהיו בסגנון `camelCase, UPPER_CASE, PascalCase`, כמשתנים וגדלים.
לדוגמה :

```
enum Day
{
    SUNDAY,
    MONDAY
};
enum Day lastDay;
```

שם הטיפוס יהיה בסגנון *camelCase*, הערכים יהיו בסגנון *PascalCase*

לדוגמא :

```
struct FeatureNode
{
    char *feature;
    struct FeatureNode *next;
};
```

3 משתנים, אופרטורים וביטויים

3.0.1 משתנים גלובליים

אין להשתמש במשתנה גלובלי או סטטי, אלא אם כן משתנה זה הוגדר כבר בקובץ הגיט שהמתרגל סיפק.

3.0.2 גדי משתנים

יש לחתה למחדר לחשב את גודלם של טיפוסים ומבנים.
משמעותו - יש להשתמש באופרטור (`sizeof`) ולא בכתיב מפורש של גודל הטיפוס או המבנה.
לדוגמא, במידה ואני רוצה להשתמש בגודל של `int`, אני לא ארשום 4, אלא (`sizeof(int)`)

3.0.3 casting

עדיף להשתמש ב-*implicit casting* על פני *explicit casting*.

3.0.4 חוקיות של מצביע

בשביל לבדוק חוקיות של מצביע, יש להשוות אותו לקבוע `NULL` ולא לערך 0.
למה? כי זה פגומים לא עובד כמו שצריך במידה ומישווים ל-0.

3.0.5 מספרי קסם

אין להשתמש במספרי קסם. (*magic numbers*). מה זה מספר קסם?

א. מוזמנים לקרוא פה: [https://en.wikipedia.org/wiki/Magic_number_\(programming\)](https://en.wikipedia.org/wiki/Magic_number_(programming))

ב. אנסה להסביר דרך דוגמאות:

דוגמה ראשונה

במידה וקיים המשימה - לכתוב תוכנית שסימירה מספרים שאנו מקבלים בקלט (בבסיס 10), למספרים בסיס 12.
מן הסתם, שאנו צריכים להשתמש במספר 12 על מנת לעשות כל מיני חישובים.

אבל אנחנו ממש לא נרצה להשתמש במספר 12 ממש.

אלא, נגיד משתנה בשם `toBase`, והערך שלו יהיה שווה ל-12.

למה? כי אם מחר המשימה משתנה להיות - המרה לבסיס 7 נניח, נוכל רק לשנות המשתנה זה, והכל יעבד לנו.

דוגמה שנייה

נניח ויש לי משימה שקשורה לחבילת קלפיים. (בלי ג'וקרים).

ידוע לי שהגודל שלו הוא 52.

וידעו לי שגם אני רוצה להתייחס לחצי מהחבילה, אז זה בתאcls 26.

אבל, אני לא אכתוב 52 ו-26 בשני מקומות נפרדים בקוד, כי זה נראה מאוד רנדומלי וככל לא קרייא.

אני כן אגדיר משתנה, נניח `deckSize`, לו אתן ערך של 52, וכאשר ארצה להתייחס לחצי מהחבילה, ארשום: `deckSize/2`.

דוגמה שלישית

כאשר משתמשים בתווים בהקשר הטקסטואלי שלהם, אין להשתמש בערך המספרי מטבלת `ascii`, אלא בקבוע טקסטואלי בתוך גרשאים.

לדוגמא, אם ארצה להשתמש בערך `ascii` של התו "a" (97), אני לא אכתוב 97, אלא "a".

3.0.6 אתחול מערך

גודל מערך בעת איתחולו יוגדר אך ורק על ידי ערך שהוא ב-`define`.
משמעותו, אסור:

`int array[100]`

ואסור :

int array[x]

כאשר x הוא משתנה.

ומותר (וזו האופציה היחידה המותרת) :

int array[SIZE]

כאשר $SIZE$ הוא גודל המוגדר ב-*.define* כמובן, הדוגמאות על מערך של *int*, אבל תקף לכל סוג מערך.

4 משפטים וזרימה

4.0.1 קוד מקוון

יש להימנע במידת האפשר מכתיבת קוד מקוון (*f*? בתוך *f* וכאן). יש לחשב איך ניתן לשנות את התנאים והלוגיקה על מנת להיצמד ככל האפשר לחלק השמאלי של המשפט.

switch case 4.0.2

כל *switch* יש לכתוב תמיד *default*, לא חייב להיות בתוכו קוד מסויים. זה בסדר להשתמש במספרים " ממש" בתנאי *switch case*.

4.0.3 שילוח גודל מערך לפונקציה

במידה ואנחנו רוצים להשתמש בגודל מערך בתוך פונקציה שהיא לא *main*, לא נשימוש *define* אליו אוחלנו את המערך, אלא נשלח את גודל המערך לפונקציה, ובתוך הפונקציה נשימוש במשתנה ששלחנו. כי"ל לערך בה מימדים.

4.0.4 שילוח מימי המערך בפונקציה

אם אנחנו רוצים, וגם אם אנחנו לא רוצים להשתמש במימי המערך בתוך פונקציה, החל ממערך דו מימי ולהלאה, הקומפיילר דרש שנשלח לו את המערך עם גדי המימדים, החל מהמימד השני. שם, אין צורך לשולח משתנה, או מספר - אלא אפשר לשים את *defines* עצם. כך לדוגמה, בימדה ויש לי מערך תלת מימי, ואני רוצה להשתמש רק בגדי המימד הראשון והשלישי שלו בתוך פונקציה, החתימה שלו תיראה כך:

```
int func(int array[][SIZE1][SIZE2], int firstDimSize, int thirdDimSize)
```

מצד אחד - אנו עונים לדרישות הקומפיילר, באמצעות *defines* שהגדנו, מצד שני - כשרצה להשתמש בגדי המימדים - אנו נשימושים במשתנים שלחנו.

4.0.5 חתימות של פונקציה

חויה להזכיר חתימות לכל הפונקציות שהשתמשו בהן בתרגילים בראש הקובץ. (מלבד *main* כמובן). זה לא משנה אם המימושים יהיו מעל או מתחת ל*main*, אבל חוות לשים חתימות, ומעלה כל המימושים האחרים. לדוגמה:

```

#include <stdio.h>

// Function prototype
int add(int a, int b);

int main() {
    int result = add(3, 4);
    printf("Result: %d\n", result);
    return 0;
}

// Function implementation
int add(int a, int b) {
    return a + b;
}

```

הגדכנו פונקציה בשם *add*, וכך שמו את חתימתה בראש הקובץ.
 הדבר נכון גם אם השתמשתנו במליאן פונקציות.
 כאשר נלמד על קבצי *h*, סעיף זה ישנה.

5 הערות

5.0.1 שפה

כל הערות יכתבו בשפה האנגלית בלבד.

5.0.2 הערת פתיחה

כל קובץ קוד שמוגש יכול בתחלתו בлок הערות לקובץ כלשהו:

```
/* ****
 * Name
 * ID
 * Assignment
 ****/
```

מספר הכוכبيות לא קריטי, אלא רק בשביל תחינה אלגנטית.

5.0.3 הערות

עליכם לתת תיאור מילולי (שוב, באנגלית) של קטעי קוד, תיאור אלגוריתם והסבירים שונים על קטעי קוד שאינם ברורים במבטו מהיר. כמוות הערות אינה מדויק, אבל כלל אכבע שייתן תשואה לגבי היקף הערות הוא שעליקם להסביר את הקוד לאדם שמכיר שפת C, אך לא מכיר את התרגילים.

5.0.4 סמנטיקת הערות

יש להשתמש בהערות בлок והערות שורה לפי הכללים הבאים:

1. כל הערות יופיעו לפני הקוד שאליו הן מתייחסות והזזהה שלחן תהיה בהתאם.
2. הערת שורה היא הערה קצרה לתיאור משתנים וקטעי קוד פשוטים. במידה וההערה גולשת לשתי שורות - יש להשתמש בהערת בлок במקום.

לדוגמה:

```
/*
This is a block comment
about the while loop
*/
while (condition)
{
    // This is a line comment about the counter
    int counter;
    /*
    This is a block
    Comment about the if condition
    */
    if (condition)
    {
        counter--;
    }
}
```

6 קריאות קוד

6.0.1 פקודה בכל שורה

יש לכתוב פקודה אחת בלבד בכל שורה.
אין להכניס שתי פקודות בשורה אחת על ידי הפרדה של ”;”.

6.0.2 גודל הזחה

גודלה של הזחה אחת יהיה *Tab* בודד.
אם אתם משתמשים בעורך שמכניס רווחים, יש להגיד את גודלה של הזחה ל4 רווחים.
אין לשלב הזחות של טאים ורווחים.

6.0.3 אורך שורה

אורך שורה לא יתרוגג מ120 תווים לרוחב.
(בהרבה סביבות עבודה ניתן להגיד את הגבלה זו באופן ייזואלי).

6.0.4 סגנון בלוקים

בפתחת בלוקים ניתן לבחור בכל אחד משני סגנונות הכתיבה הבאים, וב惟ד שאחדות הסגנון תישמר לאורך כל הקובץ.

```
if (!arr)
{
    return NULL;
}
if (!arr) {
    return NULL;
}
```