

Coding Style

מבוא למדמ"ח תשפ"ה

14 בדצמבר 2024

מסמך זה אינו המלצה, וחריגה מהכללים הכתובים במסמך זה תגרור הורדת ניקוד בתרגילי הבית. לכן, עליכם לעבוד על פי *coding style* שבמסמך זה.

1 הקדמה

בניגוד לדעות הנפוצות, רוב זמנם של מתכנתים אינו מוקדש לכתיבת קוד חדש. רוב זמנו של המתכנת מוקדש לתחזוקה, שדרוג ותיקון תקלות בקוד קיים. רוב הקוד בימינו בנוי על קטעי קוד קיימים וכל שפת תכנות מגיעה עם ספריות ופונקציות מגוונות המוכנות לשימוש. קוד שאינו כתוב בצורה פשוטה ואינו מכיל הערות והסברים בתוכו הינו קוד שיהיה מאוד קשה ומסובך לתקן, לשדרג או לתחזק בעתיד. מטרת מסמך זה היא הצגת סגנון ואופן כתיבת קוד *C* כפי שנדרש בקורס. 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.

מסמך זה מכיל דרישות נפוצות ונחשב מינימלי בהיקפו. מי שמעוניין לאמץ הרגלי 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 להתקמפל על שרתי האוניברסיטה ללא שגיאות (*errors*) או אזהרות (*warnings*). 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

1.2.1 *PascalCase*

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

1.2.2 *camelCase*

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

1.2.3 *UPPER_CASE*

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

1.2.4 *snake_case*

כתיבה באותיות קטנות בלבד, עם קו תחתון מפריד 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 וייכתב בסגנון *camelCase*.
לדוגמה - *firstName*, *playerHealth*.
משתנה שהשימוש בו קצר טווח יכול לקבל שם קצר וסתמי (x, y, i, j, k) .
לדוגמה:

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

ניתן יחסית בקלות לנחש מה הלולאה עושה.
על כל חיים שיש לשחקן, הניקוד שלו עולה ב100.
ה*i* הוא משתנה קצר טווח (ללולאה בלבד), ואילו החיים של השחקן - משתנה ארוך טווח ולכן בעל משמעות.

2.0.3 שם פונקציה

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

2.0.4 שמות פונקציות בוליאניות ומשתנים בוליאניים

שמות אלו יתחילו במילה *is* (*isValid*, *isEmpty*), כאילו מתחילים שאלה.
במקרים מסוימים, יתכן שמילה אחרת תתאים יותר - אבל עדיין יש לשמור על הרעיון של שאלה.
לדוגמה - *shouldSort*, *canEvaluate*.

2.0.5 קבועי #define

הנ"ל יהיו בסגנון *UPPER_CASE*.
לדוגמה - *MAX_INPUT*, *PI*.

2.0.6 enum

שם הטיפוס יהיה בסגנון *PascalCase*, הערכים יהיו בסגנון *UPPER_CASE*, והמשתנים בסגנון *camelCase*, כמשתנים רגילים.
לדוגמה:

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

שם הטיפוס יהיה בסגנון *PascalCase*, הערכים יהיו בסגנון *camelCase*.
לדוגמה:

```
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 גדלי משתנים

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

3.0.3 casting

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

3.0.4 חוקיות של מצביע

בשביל לבדוק חוקיות של מצביע, יש להשוות אותו לקבוע `NULL` ולא לערך 0. למה? כי זה לפעמים לא עובד כמו שצריך במידה ומשווים ל0.

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 וקיבלנו משימה - לכתוב תוכנית שממירה מספרים שאנו מקבלים כקלט (בבסיס 10), למספרים בבסיס 12. מן הסתם, שאנחנו צריכים להשתמש במספר 12 על מנת לעשות כל מיני חישובים. אבל אנחנו ממש לא נרצה להשתמש ב12 ממש.

אלא, נגדיר משתנה בשם `toBase`, והערך שלו יהיה שווה ל12. למה? כי אם מחר המשימה משתנה להיות - המרה לבסיס 7 נניח, נוכל רק לשנות משתנה זה, והכל יעבוד לנו.

דוגמה שנייה

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

ידוע לי שהגודל שלה הוא 52.

וידוע לי שאם אני רוצה להתייחס לחצי מהחבילה, אז זה בתאכלס 26.

אבל, אני לא אכתוב 52 ו26 בשני מקומות נפרדים בקוד, כי זה נראה מאוד רנדומלי וכלל לא קריא.

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 גרשיים.

לדוגמה, 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 שהוגדר ב-`define`.

משמע, אסור:

```
int array[100]
```

ואסור:

```
int array[x]
```

כאשר x הוא משתנה.

ומותר (וזה 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 מכתובת קוד מקוון (*if* בתוך *if* וכדו').
יש לחשוב איך ניתן לשנות את התנאים והלוגיקה על מנת להיצמד ככל האפשר לחלק השמאלי של המסך.

4.0.2 *switch case*

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

4.0.3 שליחת גודל מערך לפונקציה

במידה ואנחנו רוצים להשתמש בגודל מערך בתוך פונקציה שהיא לא *main*, לא נשתמש ב-*define* איתו אתחלנו את המערך, אלא נשלח את גודל המערך לפונקציה, ובתוך הפונקציה נשתמש במשתנה ששלחנו.
כנ"ל למערך במימדים.

4.0.4 שליחת מימדי המערך בפונקציה

גם אם אנחנו רוצים, וגם אם אנחנו לא רוצים להשתמש במימדי המערך בתוך פונקציה, החל ממערך דו מימדי והלאה, הקומפיילר דורש שנשלח לו את המערך עם גדלי המימדים, החל מהמימד השני.
שם, אין צורך לשלוח משתנה, או מספר - אלא אפשר לשים את ה-*defines* עצמם.
כך לדוגמה, במימדה ויש לי מערך תלת מימדי, ואני רוצה להשתמש רק בגדלי המימד הראשון והשלישי שלו בתוך פונקציה, החתימה שלה תיראה כך:

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

מצד אחד - אנו עונים לדרישות הקומפיילר, באמצעות ה-*defines* שהגדרנו, מצד שני - כשנרצה להשתמש בגדלי המימדים - אנו נשתמשים במשתנים ששלחנו.

4.0.5 חתימות של פונקציה

חובה להגדיר חתימות לכל הפונקציות שהשתמשתן בהן בתרגיל בראש הקובץ. (מלבד ה-*main* כמובן).
זה לא משנה אם המימושים יהיו מעל או מתחת ל-*main*, אבל חובה לשים חתימות, ומעל 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;
}

```

הגדרנו פונקציה בשם *add*, ולכן שמנו את חתימתה בראש הקובץ.
הדבר נכון גם אם השתמשותם במליון פונקציות.
כאשר נלמד על קבצי *h*, סעיף זה ישתנה.

5 הערות

5.0.1 שפה

כל ההערות יכתבו בשפה האנגלית בלבד.

5.0.2 הערת פתיחה

כל קובץ קוד שמוגש יכיל בתחילתו בלוק הערות לקובץ כלהלן:

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

מספר הכוכביות לא קריטי, אלא רק בשביל תחימה אלגנטית.

5.0.3 הערות

עליכם לתת תיאור מילולי (שוב, באנגלית) של קטעי קוד, תיאור אלגוריתם והסברים שונים על קטעי קוד שאינם ברורים במבט מהיר. כמות ההערות אינה מדע מדויק, אבל כלל אצבע שייתן תחושה לגבי היקף ההערות הוא שעליכם להסביר את הקוד לאדם שמכיר שפת C, אך לא מכיר את התרגיל.

5.0.4 סמנטיקת הערות

יש להשתמש בהערות בלוק והערות שורה לפי הכללים הבאים:

1. כל ההערות יופיעו לפני הקוד שאליו הן מתייחסות וההזחה שלהן תהיה בהתאם.
2. הערת שורה היא הערה קצרה לתיאור משתנים וקטעי קוד פשוטים. במידה וההערה גולשת לשתי שורות - יש להשתמש בהערת בלוק 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 להגדיר את גודלה של ההזחה ל-4 רווחים.
אין לשלב הזחות של טאבים ורווחים.

6.0.3 אורך שורה

אורך שורה לא יחרוג מ-120 תווים לרוחב.
(בהרבה סביבות עבודה ניתן להגדיר קו אנכי שיציג את ההגבלה הזו באופן ויזואלי).

6.0.4 סגנון בלוקים

בפתיחת בלוקים ניתן לבחור בכל אחד משני סגנונות הכתיבה הבאים, ובלבד שאחידות הסגנון תישמר לאורך כל הקובץ.

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

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