מבט ראשוני

מה זה C?

נקודות המכירה של C הן פשטות, מהירות, וגישה לזיכרון. C יכולה לרוץ על כמעט כל דבר, מהמחשב שלכם, לטלפון, ואפילו על הטוסטר שלכם. בגלל הפשטות שלה וגילה הוותיק אפשר לראות C כמעט בכל מקום, הדבר גרם להרבה שפות לקחת השראה מ-C ולכן אנשים שבאים מ-Java לדוגמה יראו הרבה דמיון בתחביר של השפה. בנוסף, הגישה הישירה לזיכרון גורמת לשפה להיות הבחירה הטבעית בכתיבת תוכנות שנמצאות ברמות הנמוכות של המחשב (לדוגמה מערכת הפעלה).

חשוב לציין ש-C באמת מינימלית. היא לא מספקת ניהול זיכרון, תכנות מונחה עצמים, או הרבה כלים שאנחנו רגילים אליהם בשפות אחרות. עם זאת, אין שפה יותר טובה ללמוד כדי להבין מה קורה מאחורי הקלעים במחשב.

יש לומר גם שלפרויקטים גדולים נעדיף להשתמש בשפה שמספקת קצת יותר כלים כגון C++, אבל עליה לא נדבר פה.

מאוד קל ללמוד C, לא קל להשתמש בשפה נכון

אין פה מקום לוויכוח, C היא שפה קלה (לדעתי בהרבה מקומות אפילו יותר קלה מפיתון).

אין בשפה הרבה דברים מוזרים שצריך ללמוד (ולא, ואני לא מחשיב ספרייה סטנדרטית כמשהו שצריך ללמוד) ובסך הכל השפה מספקת את האבנים הבסיסיות כדי לכתוב הכל.

לעומת זאת, לתכנת נכון ב-C זה קשה.

צריך לנהל זיכרון, צריך לתאם משתנים, פרויקטים גדולים נהיים מאוד מבולגנים מאוד מהר אם לא נזהרים.

לכן, המטרה העיקרית כאן היא ללמד את השפה, עם הרגלים טובים לעתיד.

בשביל להיות מתכנתים טובים ב-C לא מספיק לדעת את השפה, צריך לדעת עוד הרבה דברים (עיצוב, רשתות, מערכות הפעלה, אבטחה, ועוד...), ולכן נסו תמיד להישאר עם ראש פתוח לרעיונות חדשים.

רגע אבל לא כיסיתם את...

אי אפשר לכסות את הכל בספר אחד, וזו גם לא המטרה.

אם נשארות לכם שאלות אתם מוזמנים לחפש באינטרנט, אני ממליץ על הדוקומנטציה הרשמית:

<https://en.cppreference.com/w/c/language>

בואו נריץ קוד!

רגע, אבל איך?

קימפול (זה אומר להפוך את הקוד שאתם יכולים לקרוא לקוד שיכול לרוץ) C הוא תהליך מורכב שגם שנוי במחלוקת.

אתם מוזמנים לחפש באינטרנט מדריכים על התקנה של קומפיילרים שונים, אבל לצורך הפרק הזה תוכלו להשתמש בקומפיילר באינטרנט, אני ממליץ על:

[The collaborative browser based IDE - Replit](https://replit.com/)

[Online C Compiler - online editor (onlinegdb.com)](https://www.onlinegdb.com/online_c_compiler)

תבנית כללית

קובץ מלווה: [C/LanguageFundumentals/RunningCode/template.c](https://github.com/AlmogManor/Computers/blob/main/C/LanguageFundamentals/RunningCode/template.c)

לצערי אי אפשר להתחיל בהסבר על מה כל דבר בתבנית עושה. בינתיים תדעו שתוכלו לשים את הקוד שלכם איפה שכתוב printf(“Hello World!”);. אתם מוזמנים גם למחוק את השורה הזו, אך לא את השורה return 0;.

#include <stdio.h>

int main(int argc, char\*\* argv) {

printf("Hello World!");

return 0;

}

משתנים

על מה ולמה?

אפשר לחשוב על משתנים כמו קופסאות, אנחנו יכולים לשים בהם ערכים ולשלוף מהם ערכים. חידוד קטן: כל משתנה יכול להכיל רק ערך אחד, אנחנו יכולים לשנות אותו, אבל הוא יהיה הערך האחרון ששמנו.

ב-C, למשתנים יש סוגים, אפשר לשים בתוך משתנה רק דברים שמתאימים לסוג שלו (קיים גם מנגנון המרה בין סוגים).

הנה רשימה קצרה של חלק מסוגי המשתנים:

|  |  |  |  |
| --- | --- | --- | --- |
| שם | שימוש | גודל | קיבולת |
| char | אותיות / מספרים שלמים | byte | אות בודדת |
| int | מספרים שלמים | לרוב 4 bytes, יתכן לעיתים רחוקות 2 כתלות במערכת ההפעלה. | לרוב בטווח (כולל)  [-231, 231 – 1]  יתכן לעיתים רחוקות:  [-215, 215 – 1] |
| long | מספרים שלמים | 8 bytes | [-263, 263 – 1] |
| float | מספרים עשרוניים | 4 bytes | [-1.2-38, 3.438]  6 ספרות אחרי הנקודה |
| double | מספרים עשרוניים | 8 bytes | [-2.3-308, 1.7308]  10 ספרות אחרי הנקודה |
| size\_t | כתובות בזיכרון | תלוי מערכת הפעלה | לא רלוונטי |

לא קריטי לזכור את הגדלים או את סוגי המשתנים, זו רק רשימה קצרה שתוכלו לקפוץ אליה אם שכחתם משהו לרגע. רשימה מלאה אפשר למצוא איפה אם לא בויקיפדיה: [C data types - Wikipedia](https://en.wikipedia.org/wiki/C_data_types).

אוקיי, איך אני משתמש בהם?

קובץ מלווה: C/LanguageFundumentals/Variables/usage.c

שאלה מצוינת! יש שלושה דברים במהלך חייהם של משתנים (על תוחלת החיים של משתנים נדבר ביום אחר). בואו נראה דוגמה:

הכרזה – ככה אנחנו מודיעים לקומפיילר שאנחנו רוצים משתנה מסוג X בשם Y. שלב זה יכול לקרות רק פעם אחת, הכרזה על משתנה יותר מפעם אחת תגרום לשגיאה (כלומר, אם השם כבר תפוס אנחנו לא יכולים להשתמש בו שוב). כך נכריז על משתנה מסוג int בשם a:

int a;

**השמה** – כך נשים ערכים בתוך המשתנה שלנו, לדוגמה כדי לשים את הערך 5 בתוך a:

a = 5;

**קריאה** – ככה נוכל לקרוא ערכים מהמשתנים שלנו, לדוגמה כדי ליצור משתנה בשם b שמכיל את הערך של a נכתוב:

int b;

b = a;

**שימו לב!** אנחנו שמים בתוך b את הערך של a. זו לא משוואה במתמטיקה! כשנשנה את a הערך של b יישאר 5:

B = 8;

אחרי השורה הזו a עדיין 5.

**בונוס!** אפשר לשלב את שלב ההכרזה וההשמה אם אנחנו רוצים ערך התחלתי קבוע:

int a = 5;

ומה עם פעולות?

קובץ מלווה: C/LanguageFundumentals/Variables/operators.c

משתנים לא היו שימושיים ממש אם לא היינו יכולים לעשות איתם דברים נכון? אל תדאגו, הנה טבלה יפה עם כל הפעולות שאפשר לעשות על משתנים:

|  |  |
| --- | --- |
| חיבור | + |
| חיסור | - |
| כפל | \* |
| חילוק (שימו לב, חילוק של משתנים שלמים תמיד מעגל למטה) | / |
| מודולו (חילוק ושמירה של השארית במקום תוצאה) | % |
| הגדלה באחד | ++ |
| הקטנה באחד | -- |
| סוגריים (למקרה שאתם רוצים לשנות סדר פעולות חשבון) | () |
| הזזת ביטים ימינה | >> |
| הזזת ביטים שמאלה | << |
| and | & |
| or | | |
| xor | ^ |
| not | ~ |

משתנים בזמן השמה אליהם

בזמן שאנחנו מבצעים השמה לתוך משתנה, יחושב קודם כל הביטוי מימין עם הערך המקורי של המשתנה, ורק אז הערך יושם לתוך היעד, כלומר:

int a = 3;

a = a \* a + 5;

קודם יבוצע a\*a (סדר פעולות חשבון), ולאחר מכן התוצאה הזו (9) תתווסף ל-5, ורק עכשיו ישתנה הערך של המשתנה a לתוצאה הסופית, 14.

כדי לבצע השמה של אות, נקיף אותה במרכאות בודדות (אחרת איך ידעו אם אנחנו מתכוונים למשתנה a או לאות a?), דוגמה:

char letter = ‘a’

פעולות בשילוב עם השמה

בואו נגיד שאני רוצה להוסיף 5 למשתנה a. זה די קאדר לכתוב:

a = a + 5;

אל תדאגו, C נחמדה אלינו. נוכל להשתמש בכל אחד מהאופרטורים בטבלה למעלה (חוץ מ++ ו--) בשילוב עם = כדי לבצע את כל מה שמימין, ואז ביצוע הפעולה יחד עם השמה למשתנה שלנו. כלומר:

a += 3 \* 2;

יתבצע קודם 3 \* 2 ואז נוסיף את התוצאה (6) לתוך a.

המרה בין משתנים

קובץ מלווה: C/LanguageFundumentals/Variables/typeCasting.c

להמיר בין משתנים זה שימושי, נגיד ואנחנו רוצים להמיר בין משתנה עם נקודה עשרונית לאחד שמכיל רק מספרים שלמים.

קיימים הרבה