גירסה 1.00 – 6.6.2004

לולאות "כל עוד יש קלט" ועוד

Nir Adar
Email: underwar@hotmail.com
Home Page: http://underwar.livedns.co.il

אנא שלחו תיקונים והערות אל המחבר.

מבוא

מסמך זה מצטרף לשורת מסמכים שאני מפרסם בחודשים האחרונים העוסקים בלימוד שפת C למתחילים. אלו מביניכם שקראו את המסמכים הקודמים יוכלו לשים לב לחפיפה מסויימת בין המסמכים, אם כי במסמך זה, כמו בכל מסמך באתר שלי, תכנים יחודיים משלו. סידרת מסמכים אלו מהווה טיוטה למהדורה השלישית של המסמך המרכזי באתר אודות שפת C – "שפת C – היסודות". אשמח לקבל את תגובתכם לטקסט, לשמוע איזה חלקים חשובים, ועל איזה איפה לוותר במהדורה הסופית.

ניר

1. הערך המוחזר של scanf

הפונקציה scanf משמשת אותנו כדי לקרוא ערכים מהמשתמש, לדוגמא:

```
int x;
scanf("%d", &x);
```

יקרא ערך מהמשתמש ויכניס אותו לתוך המשתנה x.

scanf מאפשרת לנו לקבל מידע לגבי פעולתה – האם היא הצליחה, האם נכשלה וכו'.
מתי scanf תיכשל למשל? לדוגמא, בקוד לעיל אנחנו מצפים שהמשתמש יקיש מספר שלם. אם הוא יקיש ערך שאינו מספר, הפעולה תיכשל. מצב מיוחד של כישלון הינו "סיום הקלט" – דבר אותו נבין יותר בשיעורים הבאים. ב-Windows מקש סיום הקלט הינו Ctrl+Z.

הפונקציה scanf מחזירה את מספר המשתנים שהיא קראה בהצלחה, או 1- במקרה שנגמר הקלט.
לדוגמא, נביט בקוד הבא:

```
int x, y, z;
z = scanf("%d%d", &x, &y);
printf("%d\n", z);
```

נתעניין מה הערך של z (הערך שהפונקציה scanf החזירה) במקרים שונים:

המשתמש הקליד	הערך של z
4 9	2
34 98	2
56 hhhh	1
xxx yyy	0
^z (שזה בעצם ctrl+z)	-1

2. לולאות "כל עוד יש קלט"

סוג שימושי של לולאות הינו לולאות הקוראות כל עוד יש קלט מהמשתמש.
דוגמא ללולאות כאלו הן לולאות שמקבלות רצף מספרים ומבצעות עליו חישוב (דוגמאות מיד בהמשך).
או למשל לולאה בסגנון: כל עוד המשתמש מקליד פקודות, בצע את הפקודות הנ"ל.
בשביל לולאות כאלה, נשתמש לרוב בערך המוחזר של scanf. תנאי הלולאה יהיה "כל עוד scanf החזירה את מה שאנחנו מצפים שהיא תחזיר".
לדוגמא, עבור לולאה שקולטת מספרים שלמים אחד אחרי השני, כל עוד קיימים מספרים, נכתוב:

```
int current_element;
while (scanf("%d", &current_element) == 1)
{
	/* here we do something on the current element */
}
```

2.1. דוגמא – מציאת סכום

התוכנית הבאה מקבלת מספרים כל עוד יש קלט, ומציגה בסוף את סכומם על המסך.
אנחנו שומרים משתנה בשם sum השומר בכל רגע נתון את הסכום עד כה. בסוף הלולאה יש בידינו את סכום כל המספרים.

```
#include <stdio.h>

int main()
{
	int sum = 0;
	int cur;

	while (scanf("%d", &cur) == 1)
	{
		sum = sum + cur;
	}

	printf("The sum is %d\n", sum);

	return 0;
}
```

תרגיל:

כתוב תוכנית המקבלת מספר לא ידוע של מספרים, ובסופם היא מחשבת את הממוצע שלהם.

פתרון התרגיל:

נכתוב תוכנית הדומה לתוכנית הסכום, ובנוסף למשתנה הסכום נשמור משתנה נוסף שישמור כמה מספרים קראנו עד כה. בסוף התוכנית נבצע את החישוב (sum / numbers_counter) ונמצא את התוצאה הנכונה. נזכור לבצע casting ל-double כדי להגיד ל-C שהביטוי שאנחנו מחשבים יכול לצאת מספר ממשי ולכן יש לחשב אותו כפעולות על ממשיים.

```
#include <stdio.h>

int main()
{
	int sum = 0, numbers_counter = 0;
	int cur;

	while (scanf("%d", &cur) == 1)
	{
		sum = sum + cur;
		numbers_counter++;
	}

	printf("The average is %d\n", (double)sum / numbers_counter);

	return 0;
}
```

2.2. דוגמא – מציאת מקסימום

נכתוב תוכנית נוספת, המקבלת מספרים ומדפיסה בסוף את המספר הגדול ביותר ביניהם.

נשמור משתנה בשם max שבכל רגע לאחר אתחולו יכיל את המספר הגדול ביותר שנקרא עד כה.

בסוף קריאת המספרים נציג את ערכו.

הנחות בסגנון הנחה זו נקראים **שמורות**. (ביחיד – **שמורה**). שמורה היא הנחה שימושית שאנחנו מניחים ודואגים לכך שהיא תתקיים בכל שלב בתוכנית.

למשל – נוח לנו להניח כי המשתנה max מכיל את הערך הגדול ביותר בכל רגע. לפיכך, כל פעם שמגיע ערך חדש, נוסיף קוד שידאג לעדכן את max כדי שהשמורה אכן תתקיים.

האלגוריתם:

- קרא את המספר הראשון, וקבע אותו להיות המקסימום.
- כל עוד לא נגמר הקלט:
  - קרא את המספר הבא.
  - אם המספר שנקרא גדול מהמקסימום הנוכחי, עדכן את המקסימום להיות המספר החדש.
- הדפס את המקסימום.

נשים לב לטיפול מיוחד במספר הראשון שאנחנו קוראים. כל איבר אחרי הראשון – יש מקסימום להשוות אליו. עבור המספר הראשון אין מספרים קודמים, ולכן נוסיף קוד מיוחד האומר "קרא את המספר הראשון, וקבע אותו להיות המקסימום".

קוד התוכנית:

```
#include <stdio.h>

int main()
{
	int max, cur;

	/* read first element - to fill max with startup value */
	if (scanf("%d", &max) != 1)
	{
		printf("No input\n");
		return 0;
	}


	while (scanf("%d", &cur) == 1)
	{
		if (cur > max) max = cur;
	}

	printf("The maximum is %d\n", max);

	return 0;
}
```

2.3. דוגמא – מציאת האיבר השני הכי גדול

נביט בבעיה הבאה: עלינו לקלוט מספר לא מוגבל של מספרים, ולהדפיס בסוף הקלט מהו המספר השני הכי גדול. ניתן להניח כי המספרים שונים זה מזה.
בעייה זו מורכבת יותר מהקודמת, וקל בתחילה לטעות במציאת האלגוריתם.
דוגמא למספר רעיונות שגויים החוזרים על עצמם:

- נמצא את האיבר הכי גדול ונחזיר את המספר הקטן ממנו ב-1 (זה לא בהכרח האיבר השני הכי קטן מבין המספרים שהופיעו).
- נמצא את האיבר הכי גדול ונחזיר את הקודם לו (איך נדע מי הוא הקודם?)

איך נגיע לרעיון הנכון לפתרון הבעיה?
אם קשה לך הקורא לראות את הפתרון, דמיין את המשחק הבא:
נניח שיש מולנו שולחן בו הרבה פתקים הפוכים. על כל פתק מספר. המטרה שלנו היא למצוא את הפתק עליו רשום המספר השני הכי גדול. הפעולות המותרות: מותר לנו לקחת פתק מהשולחן ולקרוא את ערכו. לאחר מכן אנחנו יכולים לזרוק את הפתק, או לחילופין לשמור אותו בידינו. כמו כן, מותר לנו לשמור מספר מוגבל של פתקים. נניח כי מותר לנו לשמור לכל היותר 3 פתקים בבעיה זו.
כיצד נפתור בעיה כזו?
הדרך הכי טבעית היא כלהלן: ניקח את שני הפתקים הראשונים ונשמור אותם בצד.
כעת כל עוד לא סיימנו לעבור על הפתקים בשולחן, כל פעם נרים פתק אחד. אם הוא גדול יותר מלפחות אחד מהפתקים השמורים, נזרוק את הקטן מבניהם ונשמור אותו. אם הוא קטן יותר מהקטן מבין הפתקים השמורים, נזרוק אותו.

כעת נעביר את הבעיה לשפה אלגוריתמית יותר. הפתקים שאנחנו שומרים הם המשתנים שאנחנו מגדירים. הפתק שביד שלנו, זהו הערך שקראנו באיטרציה הנוכחית של הלולאה.
המטרה שלנו היא בעצם שיהיו בידינו משתנים שישמרו מה הם שני האברים הגדולים ביותר שהיו עד כה.
שאלה: מדוע לא מספיק לשמור רק את האיבר השני הכי גדול?
תשובה: כי אז היה חסר בידינו מידע לצורך השלמת המשימה. אם נקבל במקרה כזה איבר שהוא גדול יותר מהאיבר שקודם היה הכי גדול, אז האיבר שקודם היה הכי גדול הופך להיות השני הכי גדול, אולם בהצעה זו לא שמרנו אותו, ולכן לא ניתן לבצע זאת.

האלגוריתם לפתרון:

נשמור שני משתנים – אחד בשם max1 והשני בשם max2, אשר נניח (שמורה) כי הם מכילים את האיבר הכי גדול והאיבר השני הכי גדול, בהתאמה.

- קרא את שני האיברים הראשונים מהקלט. דאג שהגדול ביניהם ימוקם ב-max1 והשני מביניהם ימוקם ב-max2.
- כל עוד לא נגמר הקלט
  - קרא איבר לתוך המשתנה X
  - אם X גדול מ-max1:
    - קבע את max2 להיות max1
    - קבע את max1 להיות x
  - אחרת, אם X גדול מ-max2,
    - קבע את max2 להיות שווה ל-x
- הדפס את max2

מימוש התוכנית:

```
#include <stdio.h>

int main()
{
	int max, max2, cur, temp;


	/* read the first two values */
	if (scanf("%d%d", &max, &max2) != 2)
	{
		printf("ERROR: No input\n");
		return 0;
	}

	/* if the second value is bigger than the first one, fix the
problem */
	if (max2 > max1)
	{
		temp = max1;
		max1 = max2;
		max2 = temp;
	}



	while (scanf("%d", &cur) == 1)
	{
```

```
            if (cur > max1)
            {
                  max2 = max;
                  max = cur;
            }
            else if (cur > max2)
            {
                  max2 = cur;
            }
      }

      printf("The second maximum is %d\n", max2);

      return 0;
}
```

EOF