מבוא לתכנות מערכות

הרצאה 3

מצביעים בשפת C

מבוסס על הרצאות של אוניברסיטת חיפה, ומבוא למדעי המחשב, אורט בראודה, תעו"ן

כתובות בזיכרון - תזכורת

זיכרון המחשב מחולק לתאים, כל אחד בגודל בית אחד.

כל משתנה מאוחסן באחד או יותר תאים רצופים בזיכרון. **כתובת המשתנה היא כתובתו של התא הראשון המכיל אותו.**

		7 6 2		
	2100	2101	2102	2103

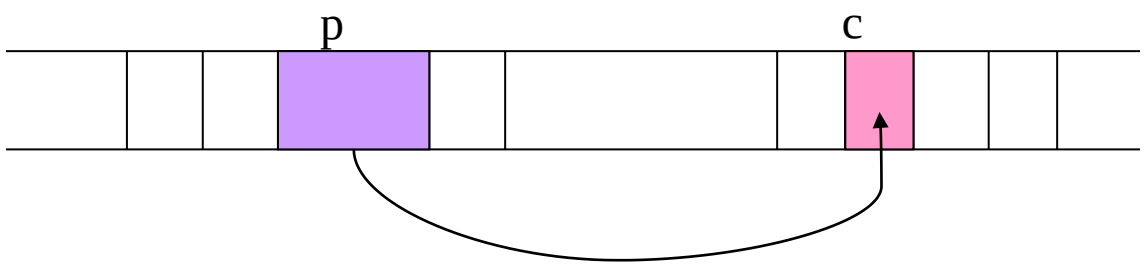
הכתובת של num היא 2100

בהנחה ש- int תופס 4 בתים

int num = 762

מצביעים (pointers)

מצביע הוא משתנה המכיל כתובת של משתנה אחר.



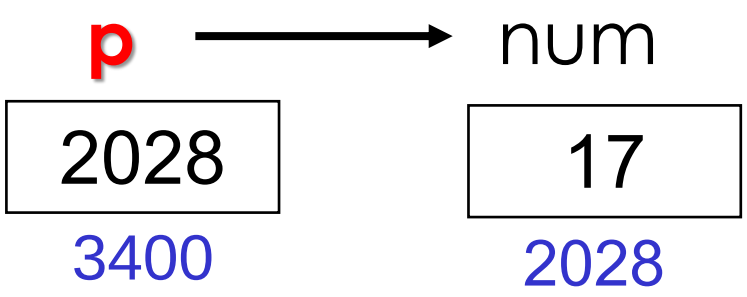
אם p הוא מצביע ל- c, אזי תוכן המשתנה p הוא הכתובת של c, כלומר p מצביע על התא ששמו הוא c.

מהו מצביע?

p הוא **מצביע** למשתנה מטיפוס **int** ששמו num.

הערך של p הוא הכתובת של num.

באמצעות p אפשר לגשת לערך של num (נראה מיד כיצד...)



הגדרה של משתנה מטיפוס מצביע

pointed_type * pointer_variable;

pointed_type: הטיפוס של הערך הנמצא בכתובת השמורה במצביע.

pointer_variable: שם המשתנה מסוג מצביע.

ישנם טיפוסי מצביעים שונים: מצביע המכיל כתובת של משתנה מסוג int , מצביע למשתנה מסוג double וכו'.

<u>דוגמאות:</u>

int *p1, *p2; (כוכבית לכל מצביע!)

double *q1, num;

מצביעים - דוגמאות

- int *ip;

  ip הוא מטיפוס int* (ip הוא מצביע ל- int)

- double *fp;

  fp הוא מטיפוס double* (fp הוא מצביע ל- double)

- double func(char *);

  הצהרה על הפונקציה func, שמקבלת מצביע ל- char ומחזירה double.

אופרטור הכתובת: &

- לכל משתנה השמור בזיכרון ניתן להצמיד את אופרטור הכתובת.
- הביטוי &a הוא כתובת המשתנה a.
- כתובת זו ניתן להשים בתוך משתנה מצביע q : q = &number

דוגמא:

```
char c;
char *p;
p=&c;
```

- שימו לב:
  - בדוגמא הנ"ל p הוא מצביע ל- char, הוא אינו char. p הוא מטיפוס char*.
  - char תופס בדיוק בית אחד בזיכרון. לא כך הדבר לגבי מצביע ל- char. כל המצביעים טופסים את אותו המקום בזכרון – 4 בתים.

אופרטור הערך המוצבע: * (כוכבית)

האופרטור באמצעותו מגיעים מן המצביע אל הערך המוצבע.

דוגמאות:

```
int a=4,x,*p_a;
p_a = &a;
x = *p_a;        /* set x to a value*/
*p_a = x/2;      /* set a to x/2*/
```

```
*p_a = *p_a + 1;      /*  ≡   a = a + 1;*/
```

שימו לב שלכוכבית יש משמעות שונה בהגדרת משתנה מסוג מצביע ובפנייה לערך המוצבע!

הקבוע NULL

- על אף שכתובות אינן נחשבות מטיפוס int, מותר לכל משתנה מסוג מצביע לקבל את הערך הקבוע 0.
- השפה מבטיחה כי 0 לעולם לא יהיה כתובת חוקית.
- הקבוע NULL מוגדר בקובץ stdio.h וערכו 0.
  (שימו לב, stdio.h כולל דברים נוספים פרט להגדרת פונקציות)
- הצבת NULL במצביע, מסמנת כי מצביע זה הוא 'ריק', כלומר אינו מכיל כתובת של משתנה :

```
int *p = NULL;
```

מצביעים – סיכום

- מצביע מכיל כתובת של משתנה מהטיפוס המתאים.
- צורת הכתיבה – type * var (עבור משתנה מטיפוס type).
- ל-* ישנם שני שימושים : בהגדרת משתנה הכוכבית מסמנת שמדובר בהגדרת מצביע. בכל מקום אחר הפנייה לכוכבית היא פנייה לערך המוצבע של מצביע.
- & הוא כתובת של משתנה. לדוגמא - &num.
- אסור להציב int לתוך מצביע.
- ניתן "לאפס" מצביע ע"י הצבת NULL - p=NULL;

תרגיל 1

מה יהיה תיאור תמונת הזיכרון של:

```
int a=1, b=2, *p1,*p2;

p1 = &a;

b = *p1;

p2 = &b;

a = a + *p1;

*p1 = *p2;
```

תרגיל 2: מה יציג קטע הקוד הבא?

```
int a;
int *p = &a;
int b;
*p = 4;
b = a + *p;
p = &b;
printf("%d  %d  %d", a, *p, b);
```

לשם מה צריך מצביעים? - דוגמה

```
void swap (int x, int y)
{
        int temp;
        temp = x;
        x = y;
        y = temp;
}

int main()
{
        int  a = 5, b = 10;
        printf("%d   %d\n", a, b);
        swap (a, b);
        printf("%d   %d\n", a, b);
        return 0;
}
```

מטרת התוכנית: להחליף בין שני ערכי קלט, ולהדפיס את ערכי המשתנים לפני ואחרי ההחלפה.

פלט התוכנית:

5 10

5 10

לא מה שרצינו...

כיצד נשנה?

- פונקציה זו אינה עושה את העבודה.
- העברנו אל swap את הערכים של a ו b,
- זוהי שיטת call by value.
- מאחר ו- x ו-y הם משתנים לוקליים של swap,
  ההחלפה התבצעה ביניהם – ללא כל השפעה על המשתנים המקוריים a ו b.
- עלינו להעביר ל swap את כתובות המשתנים!
- זוהי שיטת call by reference.

מה עושים עם זה?

- <u>ניזכר במנגנון העברת הפרמטרים לפונקציה:</u>

  הפרמטרים מועברים by value – כלומר רק הערך ולא המשתנה עצמו – ולכן השינויים שהפונקציה מבצעת אינם נשמרים במשתנה

- <u>מה יקרה אם נעביר מצביע כפרמטר?</u>

  בצורה זו אנו מאפשרים לפונקציה לגשת ישירות לזיכרון ולשנות את המשתנה המוצבע

- <u>כיצד נקרא לפונקציה?</u>

  אחד הפרמטרים יהיה type*

  בקריאה לפונקציה נרשום func(&x)

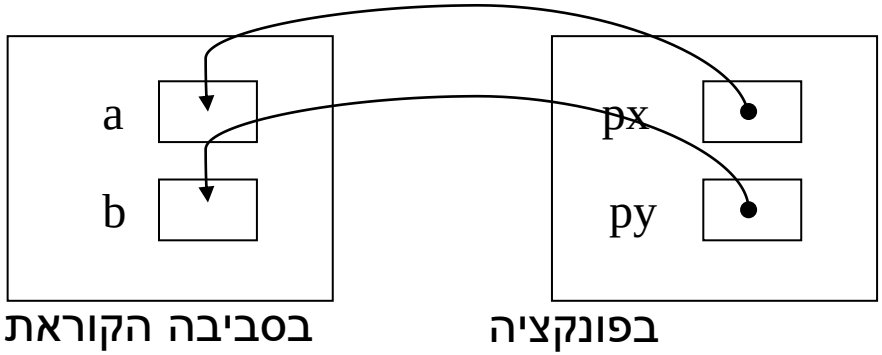
העברת פרמטרים

- באמצעות פוינטרים אפשר להעביר כתובות של משתנים מהסביבה הקוראת אל הפונקציה.

```
/* swap: CORRECT */
void swap(int *px, int *py)
{
        int temp;
        temp= *px;
       *px = *py;
       *py =temp;
}
```

```
int main()
{
        int  a = 5, b = 10;
        printf("%d   %d\n", a, b);
        swap (&a, &b);
        printf("%d   %d\n", a, b);
        return 0;
}
```

מפת הזיכרון:



סיבות נוספות לשימוש במצביעים:

- החזרה של יותר מערך אחד מפונקציה;
- העברה של מבנה נתונים "גדול" כפרמטר לפונקציה;
- גישה נוחה של מספר פונקציות לאותם משתנים (כלומר, לאותם מקומות בזיכרון);
- פעולות פשוטות יותר על מקומות וערכים בזיכרון, למשל מעבר על איברי מערך (בהמשך...)
- הקצאת זיכרון דינאמית (תוך כדי ריצת התוכנית) – בהמשך...
- יצירת קשרים בין נתונים במבני נתונים שונים – אפשר מנתון אחד להגיע לנתון נוסף הנמצא במקום אחר בזיכרון (בקורסים הבאים).

"החזרה" יותר מערך אחד מפונקציה-דוגמא נוספת

<u>תרגיל:</u>

יש לכתוב פונקציה המקבלת פרק זמן נתון בדקות ומחזירה זמן זה בשעות ודקות (ערך בין 0 ל- 59).

יש לכתוב גם את הפונקציה ראשית שקוראת לפונקציה.

"החזרה" יותר מערך אחד מפונקציה

```
#define MIN_PER_HOUR 60

void convertMinutesToHours (int time, int *pHours, int *pMinutes);

int main()
{
    int time, hours, minutes;
    printf("Enter time duration in minutes: ");
    scanf("%d", &time);
    convertMinutesToHours (time, &hours, &minutes);
    printf("\nThe time equals to %d hours and %d minutes.", hours, minutes);
    return 0;
}

void convertMinutesToHours (int time, int *pHours, int *pMinutes)
{
     *pHours = time/MIN_PER_HOUR;
     *pMinutes = time%MIN_PER_HOUR;
}
```

מצביע כפול

משתנה שמכיל כתובת של מצביע אחר הוא מצביע כפול. במילים אחרת – מצביע למצביע לנתון.

דוגמא :

```
int a, b, *pa, **ppa;
pa = &a;
ppa = &pa;
*pa = 8;
*ppa = &b;
**ppa = 6;
```

בהמשך נראה את השימוש במצביע כפול...