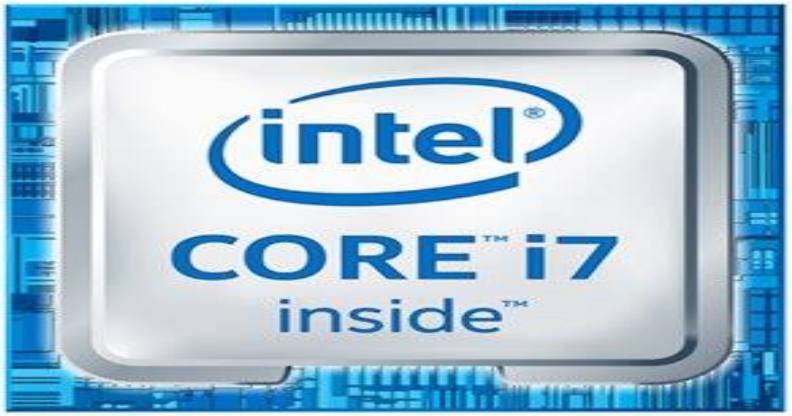
קורס יסודות התכנות בשפת C

פרק 15

קדם מעבד Preprocessor



ד"ר שייקה בילו

יועץ ומרצה בכיר למדעי המחשב וטכנולוגית מידע

מומחה למערכות מידע חינוכיות, אקדמיות ומנהליות

חזרה - ייצוג מספרים בבסיסים שונים

הקס-דצימאלי	ייצוג בינארי	דצימאלי
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

חזרה - ארגון הזיכרון לפי בתים

- תוכניות מתייחסות לכתובות 'מדומות' (וירטואליות)
- באופן אבסטרקטי:

1. מערך גדול של בתים
2. בפועל ממומש עם היררכיה שלמה של סוגי זיכרונות
3. מהדר (compiler) + רכיבי חומרה קובעים את המיפוי לזיכרון הפיזי.
4. נראה את ייצוג הכתובות במערך זה.

חזרה - ייצוג מידע

- גודל בבתים של אובייקטים שונים של C:

C Data Type	Compaq Alpha	Typical 32-bit	Intel IA32
int	4	4	4
long int	8	4	4
char	1	1	1
short	2	2	2
float	4	4	4
double	8	8	8
long double	8	8	10/12
char *	8	4	4

חזרה - לוגיקה פסוקית (בוליאנית)

- פותחה על ידי George Bool במאה ה - 19
  - נקודד 'אמת' כ – 1 ו 'שקר' כ - 0

And

&	0	1
0	0	0
1	0	1

Or

|	0	1
0	0	1
1	1	1

Not

!	
0	1
1	0

Exclusive-Or (Xor)

^	0	1
0	0	1
1	1	0

חזרה - ייצוג מספרים במחשב

- כל פריט מידע במחשב נשמר כרצף של ביטים
- ביט יכול לקבל אך ורק את הערכים 0 או 1
- רצף של שמונה ביטים מהווה בית (byte) שהיא יחידת הזיכרון הקטנה ביותר בעלת כתובת
- בשפת C מוגדרים אופרטורים לעבודה עם ביטים.
- עבודה עם ביטים נעשית על משתנים המייצגים מספרים שלמים בלבד!!! (char, short, int, long)
  - ניתן להשתמש גם בגרסת המשתנה ה-unsigned של כל אחד מהם.

חזרה - ייצוג מספרים בבסיסים שונים

- מספרים במחשב מיוצגים על בסיס 2 (בסיס בינארי)
- ניתן להתייחס למספר בייצוג אוקטלי (בסיס 8) ע"י הוספת 0 לפני המספר.
- לדוגמה המספר 042 באוקטלי מייצג את המספר 34 בבסיס עשרוני
- תו המרה רלבנטי הוא o%

חזרה - פעולות חישוב על ביטים

- ניתן להתייחס למספר בייצוג הקס-דצימלי (בסיס 16) ע"י הוספת 0x לפני המספר. לדוגמה המספר 0x42 מייצג את המספר 66 בבסיס עשרוני.
  - תו המרה רלבנטי הוא %x (או %X לקבלת אותיות גדולות)
- על מנת להשתמש באופרטורים של ביטים נדרש לייצג את המספר בייצוג בינארי
- פעולות חישוב על ביטים מבוצעות ביט מול ביט
- בדרך כלל עדיף להשתמש בגרסת **unsigned** של המשתנה

חזרה - המרה מבינארי לעשרוני

- אלגוריתם המרה של מספר בבסיס בינארי למספר בבסיס עשרוני:
  - לכל ספרה בייצוג יש "משקל" שהוא 2 בחזקת האינדקס של מיקומה במספר (אינדקס מתחיל מימין ומהערך 0)

אינדקס	0	1	2	3	4	5	6	7	8
משקל	1	2	4	8	16	32	64	128	256
חישוב	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8

  - כפול כל ספרה במשקלה המתאים עפ"י מיקומה בייצוג הבינארי, הערך של המספר העשרוני הוא סכום המכפלות

חזרה - המרה מבינארי לעשרוני

- לדוגמה, הייצוג הבינארי הבא:

00001101

- המיקומים הם:

128=0 ,64 =0 , 32=0 ,16=0 ,8=1 ,4=1 ,2=0 ,1=1

- הייצוג הנ"ל מייצג את המספר 13:

1*1+0*2+1*4+1*8+0*16+0*32+0*64+0*128 =

1*1 + 1*4 + 1*8 =

1+4+8 = 13

חזרה - המרה מעשרוני לבינארי

- אלגוריתם המרה של מספר בבסיס עשרוני למספר בבסיס בינארי:
  - כל עוד המספר אינו 0
  - חלק את המספר ב-2
  - שמור את השארית
- הייצוג הבינארי של המספר, משמאל לימין, הוא אוסף השאריות מהסוף להתחלה.
- כאשר יש מצב שחלק מהספרות חסרות, הן מקבלות את הערך 0.

חזרה - המרה מעשרוני לבינארי

- לדוגמה – המספר 8 בייצוג בינארי הוא: 00001000

מספר	שארית
8	0
4	0
2	0
1	1
0	

חזרה - רשימת האופרטורים

דוגמא	הסבר	משמעות אופרטור	סימן
2&3=2	ביט התוצאה יהיה 1 אם ורק אם שני הביטים המקבילים בפעולה שווים ל-1	AND	&
2|3=3	ביט התוצאה יהיה 1 אם לפחות אחד משני הביטים המקבילים בפעולה שווה ל-1	OR	|
2^3=1	ביט התוצאה יהיה 1 אם ורק אם בדיוק אחד משני הביטים המקבילים בפעולה שווה ל-1 (כלומר הביטים שונים זה מזה)	XOR	^

- ניתן להשתמש באופרטורים הבאים לקיצור: &= , |= , ^=

חזרה - רשימת האופרטורים

דוגמא	הסבר	משמעות אופרטור	סימן
2<<1=4	מזיז את הביטים של משתנה המקור (האופרנד השמאלי) מספר צעדים שמאלה (עפ"י ערך האופרנד מימין). הפעולה מאפסת את התאים שנותרו ריקים מימין. הפעולה שקולה לכפל ב-2.	Shift Left	<<
2>>1=1	מזיז את הביטים של משתנה המקור (האופרנד השמאלי) מספר צעדים ימינה (עפ"י ערך האופרנד מימין). אלגוריתם מילוי התאים הריקים הוא תלוי מערכת. הפעולה שקולה לחלוקה ב-2.	Shift Right	>>
!2=253	מחליף את ערך הביט: 1 הופך ל-0 ו-0 הופך ל-1	Complement	!

חזרה - טבלאות פעולה

XOR

Bit 1	Bit 2	^ (xor)
0	0	0
0	1	1
1	0	1
1	1	0

OR

Bit 1	Bit 2	| (or)
0	0	0
0	1	1
1	0	1
1	1	1

AND

Bit 1	Bit 2	& (and)
0	0	0
0	1	0
1	0	0
1	1	1

שאלות על השעור הקודם?

קדם מעבד – נושאי לימוד

- פקודות נפוצות
  - הוספת קבצים - #include
  - הגדרת קבועים - #define
  - ביטול הגדרת קבועים #undef
  - הגדרת פקודות macro
  - הידור מותנה
  - התניה של פקודות קדם-עיבוד וקומפילציה
  - פקודות קדם מעבד נוספות

קדם מעבד – נושאי לימוד

Preprocessor directive	
File inclusion directive	#include
Macro substitution directive	#define
Undefine symbol directive	#undef
Conditional directive	#if , #elif , #else , #endif , #ifdef , #ifndef
Miscellaneous directive	#pragma , #line , #error
Operator in preprocessor	# , ##, define( )

קדם מעבד - Pre-Compiler

- קדם-מעבד (Pre-Processor / Pre-Compiler) הוא הכלי שעובר על התוכנית לפני שלב ההידור (קומפילציה)
- לקדם-מעבד שפה משלו והפקודות שלו עוסקות בעריכת טקסט קובץ המקור כהכנה לשלב ההידור
- פקודות הקדם-מעבד משנות את תוכן הקובץ ומחליפות תווים בקובץ ללא התייחסות למשמעות התווים בשפת C

הכללת קבצים - #include

- פקודת #include נועדה להכללת קבצים
- שורת ה-#include מוחלפת בתוכן הקובץ המוכלל
- בקבצים המוכללים נהוג להצהיר על קבועים, משתנים ופונקציות גלובליות.
- לפקודה 2 צורות כתיבה:

1. #include <filename> - חיפוש הקובץ נעשה בתיקיות מוגדרות מראש
2. #include "filename" - חיפוש הקובץ נעשה קודם כל בתיקייה הנוכחית (בה נמצא קובץ המקור) ורק אח"כ בתיקיות מוגדרות.
   - ניתן לפרט מסלול מלא למיקום הקובץ

דוגמאות #include


```
#include <stdio.h>

#include “DefinitionFile.h”

#include “c:\\HeaderFiles\\DefinitionFile.h”

#include “..\\someDirectory\\DefinitionFile.h”
```

החלפת תווים (מאקרו) - #define

- פקודת #define משמשת להחלפת חלקי טקסט בקובץ
- מבנה הפקודה

#define identifier alternative-text

- כל מופע של identifier יוחלף ב-alternative-text
- identifier – אוסף תווים המהווים גם שם משתנה חוקי (מתחיל בתו ומכיל רק אותיות וספרות). נהוג שימוש באותיות גדולות.

החלפת תווים (מאקרו) - #define

- alternative-text – כל טקסט שהוא (ברירת המחדל היא שורה בודדת אך ניתן להוסיף \ בסוף שורה לאיחוד שורות)
- identifier מוכר משורת ההגדרה ועד סוף הקובץ
- ההגדרה יכולה להשתמש בהגדרות קודמות
- החלפה מתבצעת רק על מופע שלם של identifier
- לא מתבצעת החלפה על מופע identifier בתוך מחרוזת (בתוך מרכאות" ")

דוגמאות שימוש - #define


```
#define OK 1
printf("OK");        \\ לא יוחלף
int NOT_OK;          \\ טקסט חלקי לא מוחלף
int status = OK;     \\יוחלף
#define INFINITE for(;;)
INFINITE printf("Hello");                מה יקרה במצב כזה?
#define DOUBLE    + NUM + NUM
#define QUADROUPLE    DOUBLE    DOUBLE
int NUM=5;
if(DOUBLE DOUBLE== 20)
   printf("Nice!");
```

מאקרו עם פרמטרים - #define

- ניתן להגדיר מאקרו שמקבל ארגומנטים כך שבזמן ההחלפה, יושמו ערכים שונים בקריאות השונות למאקרו
- מבנה הפקודה

#define identifier(argument-list) alternative-text

- חשוב להקפיד שה-) יהיה צמוד לשם identifier
- argument-list רשימת מזהים מופרדת בפסיקים
- כל מופע של מזהה-ארגומנט בתוך ה- alternative-text יוחלף בערך הנשלח בזמן הקריאה

```
#define max(A,B) ((A)>(B)? (A) : (B))
x=max(a+b, c+d); → x=((a+b)>(c+d)?(a+b) : (c+d));
```

מאקרו עם פרמטרים - #define

- יתרונות
  - מאקרו עובד עבור כל סוג משתנה
  - inline - הקוד מתפתח ע"י החלפת טקסט ללא קריאה לפונקציה
- חסרונות
  - ערכי A ו-B בדוגמה הנ"ל מחושבים שוב ושוב בעוד שבקריאה לפונקציה יחושבו פעם אחת
  - max(i++, j++); → יגרום להגדלות כפולות
  - נדרשת תשומת לב לשימוש נכון בסוגריים
- דוגמאות נוספות
  - פעולת החלפה swap(t, a, b)

ביטול הגדרה - #undef

- ניתן לבטל מאקרו והגדרות שהוגדרו ע"י שימוש בפקודת ה- define
- השימוש נועד על מנת לוודא שאין כפילות הגדרה
- מבנה הפקודה:

```
#undef identifier
```

שאלות?

Conditional Inclusion - הידור מותנה

- ניתן לשלוט בפעולות הקדם-מעבד ע"י שימוש ב-
- conditional-statements
- תנאים אלה מתפתחים ומחושבים בזמן קומפילציה (הידור)
- בעזרת שיטה זו ניתן להכליל/להשמיט קוד בזמן הקומפילציה
- פקודת #if מחשבת ערך של קבוע מסוג integer. אם ערך הביטוי שונה מ-0, כל הוראות הקוד עד #elif/#else/#endif הקרוב יוכללו בקובץ
- בביטוי מותר לכלול כל אופרטור C הפועל על מספרים שלמים
- בביטוי אסור השימוש ב-casting, sizeof ו-enum

הידור מותנה – מבנה הפקודה


```
#if integer-expr1
#elif integer-expr2
#else
#endif
```

- ההוראות המוכללות יכולות להיות פקודות #include, הצהרות והגדרות של משתנים פונקציות ועוד...

הידור מותנה - שילוב עם #defined

- הביטוי defined(name) יחזיר 1 אם name הוגדר קודם לכן ע"י #define ו-0 אחרת
  - אין חשיבות לערך שניתן בזמן ההגדרה אלא רק לעצם ההגדרה
- ניתן להשתמש בפקודה על מנת למנוע הכללה חוזרת של קבצים
- דוגמה למעטפת לקובץ header כלשהו

```
#if !defined(INC_FILE)
#define INC_FILE          // יקרה בפעם הראשונה בלבד
        // header file content
#endif
```

הידור מותנה - שימושים נוספים

- שימוש נפוץ בשילוב פקודות #if ו-#defined הוא לצרכי ניפוי שגיאות (debugging)
- דוגמה:

```
#if defined(DEBUG)
   printf("Variable x has value = %d", x);
#endif
```

הידור מותנה - קיצור ע"י #ifdef

- כיוון שהשימוש בביטוי תנאי להגדרות #define נפוץ, הוגדר עבורו קיצור:

```
#ifndef INC_FILE
#define INC_FILE
        // header file content
#endif
```

פקודות נוספות

- קבועים נוספים מוגדרים מראש
  - LINE - מספר השורה בקובץ הנוכחי
  - FILE - שם הקובץ הנוכחי
  - DATE - התאריך בו הקובץ הנוכחי עבר הידור
  - TIME - הזמן בו הקובץ הנוכחי עבר הידור

פקודות נוספות

- #line number – מאתחלת את ערך מספרי השורות בתכנית ל-number.
  - ניתן גם להוסיף במרכאות " " שם קובץ חלופי לשם הקובץ הנוכחי
- #error error-text – פקודת קדם מעבד המייצרת הודעת שגיאה בתהליך הקומפילציה
- assert(condition) - פקודה (המוגדרת ב-"assert.h") המדפיסה הודעת שגיאה ומפסיקה את התכנית כאשר התנאי אינו מתקיים, הודעת השגיאה תכלול את התנאי הרלבנטי, שם הקובץ ומספר השורה.

שאלות?

תרגילים

תרגילים

1. הגדר מאקרו המקבל כארגומנט את רדיוס המעגל ומחשב את שטח מעגל והיקפו.
2. הגדר מאקרו בשם MINIMUM2 המקבל 2 מספרים ומחזיר את הקטן מביניהם. הגדר שני משתנים גלובליים בעלי ערך שלם והשתמש במאקרו להדפסת המספר הקטן מביניהם.
3. הגדר מאקרו MINIMUM3 המחזיר את הערך הקטן מבין 3 מספרים. ניתן להשתמש במאקרו מהשאלה הקודמת.
4. הגדר מאקרו בשם PRINTARRAY המדפיס מערך של מספרים שלמים. המאקרו מקבל את המערך ואת גודלו.
5. כתוב מאקרו בשם DIVBYTEN שתפקידו לבדוק האם מספר שלם מתחלק ב- 10 ללא שארית אז ערכו 1 אחרת ערכו 0.

תרגילים

6. הגדר מאקרו בשם ISOCTAL הבודק האם מספר שנקלט הוא מספר בבסיס 8, אם כן המאקרו שווה 1 אחרת הוא שווה 0.
7. הגדר מאקרו בשם ISDECIMAL הבודק האם מספר שנקלט הוא מספר בבסיס 10, אם כן המאקרו שווה 1 אחרת הוא שווה 0.
8. הגדר מאקרו בשם ISHEXADECIMAL הבודק האם מספר שלם שנקלט הוא מספר בבסיס 16, אם כן המאקרו שווה 1 אחרת הוא שווה 0.
9. הגדר מאקרו בשם SWAP המבצע החלפה בין שני משתנים ללא שימוש בתא עזר.
10. כתבו מאקרו בשם DETAILS המציג את פרטי קובץ התוכנית כגון שם הקובץ, התאריך והשעה שבה נוצר.

קדם מעבד – סיכום

- פקודות נפוצות בשימוש בקדם מעבד בשפת C:
  - הוספת קבצים - #include
  - הגדרת קבועים - #define
  - ביטול הגדרת קבועים #undef
  - הגדרת פקודות macro
  - הידור מותנה #if
  - התניה של פקודות קדם-עיבוד וקומפילציה
  - פקודות קדם מעבד נוספות – LINE, DATE, TIME, FILE

שאלות?