3. יחידת ספריה ופרויקט ב- C

יחידת הלימוד "**עיצוב תוכנה**" עוסקת בנושא הפיתוח והעיצוב של מערכות. חלק מעקרונות הפיתוח הם: תכנון מהפרט אל הכלל, חלוקה למודולים ועבודת צוותים על פי ממשקים ברורים.

מערכת אחת יכולה להיות מורכבת מכמה מודולים, שכל אחד מהם מהווה יחידה עצמאית ובלתי תלויה ביחידות האחרות. כל מודול מורכב מאוסף של פעולות עצמאיות אף הן. מודול אחד עשוי לשמש מספר תכניות שכל אחת מהן מטפלת במערכת אחרת.

כל מודול המטפל בנושא אחר נקרא "**יחידת ספריה**".

ניתן לכתוב יחידת ספריה כקובץ עצמאי הכולל את כל הפונקציות של המודול. יחידת ספריה תישמר כקובץ c. הקישור ליחידת הספריה נעשה במשפט include מהתכנית המפעילה את יחידת הספריה. מודול יכול להיות מקושר לתכנית או למודול אחר.

שלבי העבודה:

1. כתוב את יחידת הספריה <u>ללא</u> פונקצייה main.
2. שמור את יחידת הספריה כקובץ שפת C.
   (לדוגמא: יחידת הספריה my_unit.c)
3. כתוב את התכנית הראשית. הקפד להוסיף את המשפט: **#include "my_unit.c"**
4. הרץ את התכנית הראשית.

שלבים ביצירת פרויקט:

1. צור קובץ כותר header file, בעל סיומת h, שיכיל את כל ההצהרות וה- include של היחידה (חלק ה- prototype). לדוגמא: my_unit.**h** .

   קובץ הכותר יכיל רק את הפונקציות הגלויות - אלו שכותב היחידה מעוניין שיעמדו לרשות המתכנת המשתמש ביחידת הספריה.

2. צור קובץ מקור source file, בעל סיומת c או cpp, שיכיל את מימוש הפונקציות שבקובץ הכותר.

   אם דרושות פונקציות עזר, הן תיכתבנה ותמומשנה בקובץ זה. פונקציות העזר הן פונקציות המוסתרות מהמשתמש ביחידה.אין הוא יודע על קיומן, ולכן גם אינו יכול להשתמש בהן.השימוש בפונקציות העזר שמור לפונקציות הגלויות.

   דוגמא לפונקציית עזר: swap - פונקציה המחליפה תוכן של שני תאי מערך בפעולה גלוייה של מיון המערך. המתכנת יכול למיין את המערך אך אינו יכול להשתמש ב- swap.

3. כתוב את התכנית הראשית. התכנית תכיל קישור לקובץ הכותר:
#include "my_unit.h"

4. יש לכלול קובץ תיעוד עברי כמקובל ליחידות ספריה ב"עיצוב תכנה".

הדגשים:

- יחידת הספריה תישמר בתיקייה בה נשמרת התכנית הראשית. אם שומרים את יחידות הספריה בתיקייה אחרת, יש לציין את נתיב החיפוש המלא הכולל שם כונן ותיקיות החל מתיקיית השורש - #include "full-path\my_unit.c"
- הקבצים יכולים להיות קבצי c או קבצי cpp.
- הפונקצייה main יכולה להופיע רק פעם אחת בכל הפרויקט, ולכן היא תכתב בתכנית הראשית בלבד.
- בתכנית הראשית מפעילים את הפונקציות של יחידת הספריה כאילו היו פונקציות מובנות בשפה. המשמעות - אסור להגדיר או לממש את הפונקציה - **שוב** - בתכנית הראשית.
- אם התכנית הכללית מקושרת ליותר מיחידת ספריה אחת, ויחידות אלו מקושרות האחת לאחרת במשפטי include, עלולה להיווצר בעיה של כפילות פונקציות ו/או של הגדרת מבני נתונים חדשים. לשם כך נשתול בקובץ הכותר הוראות של הקדם-מעבד (pre-proccessor) כמוגדם בתרגיל שבעמוד הבא.
- בכל פעם שתוכלל יחידת הספריה, יבדוק המעבד: "אם טרם הוכלל קובץ זה בפרויקט, יש להכלילו עתה".

//-------- alpha.h --- Header file ----------
#ifndef ALPHA_H
#define ALPHA_H
#include <stdio.h>
void alpha_one ();
void alpha_two ();
#endif
	//--------- alpha.c --- source file ----------
	void alpha_one ()
	{
	printf ("alpha_one ");
	}
	void alpha_two ()
	{
	printf ("alpha_two ");
	}

//---------- beta.h --- Header file ---------
#ifndef BETA_H
#define BETA_H
#include <stdio.h>
#include "alpha.h"
#include "gamma.h"
void beta_one ();
void beta_two ();
void beta_three ();
#endif
	//---------- beta.c --- source file ----------
	void beta_one ()
	{
	printf ("beta_one \n");
	alpha_one ();
	printf (" from beta \n");
	}
	void beta_two ()
	{
	printf ("beta_two \n");
	gamma_two ();
	printf (" from beta \n");
	}
	void beta_three ()
	{
	printf ("beta_three \n");
	}

//-------- gamma.h --- Header file --------
#ifndef GAMMA_H
#define GAMMA_H
#include <stdio.h>
#include "alpha.h"
void gamma_one ();
void gamma_two ();
#endif
	//-------- gamma.c --- source file --------
	void gamma_one ()
	{
	printf ("gamma_one \n");
	}
	void gamma_two ()
	{
	printf ("gamma_two \n");
	alpha_one ();
	printf (" from gamma \n");
	}

//---------- chk_unit.cpp ----------
//-------- the main program includes other units -----
 	
#include "beta.h"
#include "gamma.h"
 	
void main ()
{
beta_one ();
beta_two ();
beta_three ();
 	
gamma_one ();
gamma_two ();
 	
}
	פלט התכנית:
	
	beta_one
	alpha_one from beta
	beta_two
	gamma_two
	alpha_one from gamma
	from beta
	beta_three
	gamma_one
	gamma_two
	alpha_one from gamma

הרצת הפרויקט בסביבת C ל- dos:

(1) **פתח פרויקט חדש**: Project → Open project …
תן שם לפרויקט. שם הפרויקט יקבל סיומת: prj

(2) **סימון הקבצים המשתתפים בפרויקט**: Project → Add item …
יש לבחור את קבצי המקור של יחידות הספריה ואת קובץ התכנית הראשית (קבצי c).

(3) **בניית קובץ ריצה**: Compile → Build all
אם בניית הפרויקט הצליחה, תופיע ההודעה Success בדומה להודעה המופיעה בעקבות קומפילציה מוצלחת.

בשלב זה ניתן להריץ את התכנית הראשית בפקודת Run (או לבצע שורה אחר שורה בהוראת Trace).

(4) בסיום העבודה - סגור את הפרויקט. Project → Close project

דוגמא לפרויקט:

my_date - יחידת ספריה המטפלת בחישובים על תאריכים.

I. קבועים וטיפוסים:

הקבועים והטיפוסים בהם עושה יחידת הספריה שימוש.

הוגדר טיפוס נתונים מופשט חדש: **טיפוס-תאריך** הכולל את השדות: יום, חודש, שנה.

II. פעולות היחידה:

הפעולות הגלויות שביחידה - ומופיעות בקובץ הכותר my_date.h

א. **קלט-תאריך** - פעולה הקולטת תאריך מן המשתמש.

ב. **הצגת-תאריך** - פעולה המקבלת תאריך ומציגה אותו על המסך.

ג. **מרחק-בין-שני-תאריכים** - פעולה המקבלת שני תאריכים ומחזירה את מספר הימים שעבר בין התאריך המוקדם יותר לתאריך המאוחר יותר. מספר ימים זה מוגדר כ"מרחק" שבין התאריכים.

ד. **חישוב-תאריך-חדש** - פעולה המקבלת תאריך ומספר שלם k, ומחזירה תאריך חדש שהוא התאריך המחושב k ימים מהתאריך שהתקבל.

הפעולות המוסתרות / פעולות העזר שביחידה - מתוארות בקובץ המקור (בלבד) my_date.c:

ה. **תקינות-תאריך** - פעולה המקבלת תאריך ומחזירה תשובה 'אמת' אם התאריך תקין ו-'שקר' אחרת.

דוגמא לתאריך לא תקין: חודש השונה ממספר החודשים הקיים בשנה, יום שאינו בתחום הימים המקובל באותו חודש. לא נבדקה תקינות תאריך לשנה מעוברת. הפעולה משמשת את הפעולה **קלט-תאריך**.

ו. **מתאריך-למספר** - פעולה המקבלת תאריך תקין ומחזירה את מספר הימים שחלף מתאריך עוגן מסויים עד לתאריך זה.

תאריך העוגן הוא תאריך הקודם לשני התאריכים (למשל 1.1.1980). כל תאריך מתורגם למספר המייצג את מספר הימים שחלף מתאריך העוגן לתאריך זה.

ז. **ממספר-לתאריך** - פעולה המקבלת מספר ומחזירה תאריך. התאריך הוא התאריך שיתקבל מהוספת מספר הימים הנתון לתאריך העוגן.

התכנית הראשית chk_date.c מפעילה את יחידת הספריה my_date על ידי הכללת קובץ הכותר של היחידה.

```
/*~~~
```

```
/*~~~
```

```
date_type new_date_calculation (date_type d, int k)
 {
        long num;
      num = date_to_num (d);
      num = num + k;
      return ( num_to_date (num));
 }
```

//~~~~~~ מימוש הפונקציות הפנימיות ~~~~~~

```
//--- פעולה המקבלת תאריך ומחזירה 'אמת' אם התאריך תקין, ו-'שקר' אחרת ---
int check_date (date_type d)
 {
        int month [13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
        //--- בדיקת שנה מעוברת. שנה מעוברת היא שנה המתחלקת ב-4 ולא מתחלקת ב-100 ---
        //--- או שנה המתחלקת ב-400. ---
        if ((d.yy%4 = = 0) && (d.yy%100 !=0) || (d.yy%400 = = 0))
                m[2] = 29;
        if ((d.mm <= 0) || (d.mm > 12))
          {
                printf ("Erorr In Month! Please Insert Month Between 1 to 12 \n");
                return FALSE;
          }
      if ((d.dd < 1) || (d.dd > month [d.mm]))
        {
            printf("Erorr In Day! Please Insert Day Between 1 to %d \n ",m[d.mm]);
            return FALSE ;
        }
       return TRUE;
 }


//--- פעולה המקבלת תאריך ומחזירה את מספר הימים שחלף מתאריך עוגן עד לתאריך זה ---
//--- הנחה: התאריך תקין. ---
long date_to_num (date d)
 {
        int month [13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
        int sum = 0 , i;
```

```
        if ((d.yy%4 = = 0) && (d.yy%100 !=0) || (d.yy%400 = = 0))
                m[2] = 29;

        //--- הוספת שנים לסכום ---
        sum = (int) fabs ((d.yy - YEAR) * 365.25);

        //--- הוספת חודשים לסכום ---
        for (i = 1 ; i < d.mm ; i++)
                sum = sum + month [i];

        //--- הוספת הימים לסכום ---
        sum = sum + d.dd;

        return (sum);
  }

//--- פעולה המקבלת מספר ומחזירה תאריך. התאריך הוא התאריך שיתקבל מהוספת ---
//--- מספר הימים הנתון לתאריך העוגן. ---
//--- הנחות: התאריך מאותחל ותקין, המספר חיובי. ---
date num_to_date (long num)
  {
        int month [13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
        date_type d;
        int temp;

        if ((d.yy%4 = = 0) && (d.yy%100 !=0) || (d.yy%400 = = 0))
                m[2] = 29;

        //--- חישוב שנה ---
        temp = (int) (num / 365.25);
        d.yy = temp + YEAR;
        num = num - temp * 365.25;

        //--- חישוב חודש ---
        d.mm = 1;
        while (num > month [d.mm])
          {
                num = num - month [d.mm];
                d.mm++;
          }

        //--- חישוב יום ---
        d.dd = num;

        return (d);
  }
```

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                chk_date.c
                main program
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
#include<stdio.h>
#include<math.h>
#include "my_date.h"

void main ()
 {
        int num;
        date d1, d2, d3;

        printf ("Enter 1st date: ");       d1 = get_date();
        printf ("Enter 2nd date: ");     d2 = get_date();

        num =  distance_between_dates (d1, d2) ;

        printf ("The Difference Between ");
        show_date (d1);
        printf (" And ");
        show_date (d2);
        printf (" Is: %d days. \n",num);
        printf ("\n");

        d3 = new_date_calculation (d2, num);

        printf ("The date in %d days will be: ", num);
        show_date (d3);
        printf ("\n");
 }
```