**רשימה מקושרת- LinkedList**

**רשימה מקושרת:** רשימת שנוצרת מהקצאות זכרון של מבנים כאשר בכל מבנה יש מאפיין מסוג פוינטר next)) שמצביע על כתובת המבנה הבא.

הפוינטר של המבנה האחרון מצביע על null.

כך נוצרת רשימה מקושרת שנבנית מהקצאות זיכרון של מבנים.

**יש להגדיר משתנים ולהצהיר על אופיין מחוץ לmain כך שיהיו גלובאליים:**

struct Item {

int val;

struct Item\* next;

};

struct Item\* head = NULL;

void freeLinkedList();

**בנוסף למבנה הגלובאלי ופוינטר head יש להגדיר עוד שני משתנים של מצביע נוכחי ומצביע קודם כך שנוכל לקשר את הרשימה:**

struct Item\* currentItem;

struct Item\* prevItem = NULL;

**דוגמא להקדמת וקישור הרשימה להקצאות של מבנים הבאים- קוד זה ירשם בפונקצית while לפי צרכי התכנית:**

currentItem = malloc(sizeof(struct Item));

if (head == NULL)

{

head = currentItem;

}

else

{

prevItem->next = currentItem;

}

currentItem->next = NULL;

prevItem = currentItem;

**פונקצייה המשחררת את הקצאות הזכרון:** רצה מהראש ועד הnext האחרון שמצביע על null

ומשחרר כל הקצאה עד סיום הרשימה.

void **freeLinkedList()**

{

struct Item\* currentItem = head;

struct Item\* releaseItem = head;

while (currentItem != NULL)

{

releaseItem = currentItem;

currentItem = currentItem->next;

free(releaseItem);

}

}

**פונקציות שימושיות**

* עדכון ערך
* חיפוש מבנה באמצעות ערך והחזרת מצביע
* בניית רשימה ממוינת של x פריטים שערכיהם רנדומליים
* הוספת פריט אחד שערכו רנדומלי

void generateList()

void generateXList(int x)

struct Item\* findVal(int val)

struct Item\* changeVal(int oldVal, int newVal)

**הקוד עצמו:**

void generateList()

{

    int ranNum;

    struct Item\* currentItem;

    struct Item\* prevItem = NULL;

    head = NULL;

    for (int i = 0; i < 10; i++)

    {

        ranNum = rand();

        currentItem = malloc(sizeof(struct Item));

        if (head == NULL)

        {

            head = currentItem;

        }

        else

        {

            prevItem->next = currentItem;

        }

        currentItem->val = ranNum;

        currentItem->next = NULL;

        prevItem = currentItem;

    }

    printList();

    //freeLinkedList();

}

void generateXList(int x)

{

    int ranNum;

    struct Item\* currentItem2;

    struct Item\* prevItem2 = NULL;

    head = NULL;

    for (int i = 0; i < x; i++)

    {

        ranNum = rand();

        currentItem2 = malloc(sizeof(struct Item));

        if (head == NULL)

        {

            head = currentItem2;

        }

        else

        {

            prevItem2->next = currentItem2;

        }

        currentItem2->val = ranNum;

        currentItem2->next = NULL;

        prevItem2 = currentItem2;

    }

    printList();

    freeLinkedList();

}

struct Item\* findVal(int val)

{

    struct Item\* currentItem = head;

    while (currentItem != NULL)

    {

        if (currentItem->val == val)

        {

            return currentItem;

        }

        currentItem = currentItem->next;

    }

}

struct Item\* changeVal(int oldVal, int newVal)

{

    struct Item\* currentItem = head;

    while (currentItem != NULL)

    {

        if (currentItem->val == oldVal)

        {

            currentItem->val = newVal;

            return currentItem;

        }

        currentItem = currentItem->next;

    }

}