**דוח מעבדה תרגיל בית 3**

**שמות מגישים:**

סער ויקטור – 312392822

אילון בן סימון – 312162951

קוד התכנית:

קובץ header:

#ifndef example

#define example

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

typedef struct SET\* PSET;

typedef struct node\* pnode;

typedef enum {FALSE,TRUE} BOOL;

void FreeSet(PSET s);

void Init(PSET\* s, int(\*isequal)(void\*, void\*), int(\*isgreater)(void\*, void\*), void(\*print)(void\*), void(\*delete)(void\*));

void createSet(void\* variable, PSET\* s);

void Err\_Msg(char\*s);

void addToSet(PSET\* s, void \*Element);

BOOL deleteFromSet(PSET\* s, void\* Element);

BOOL findInSet(PSET\* s, void\* Element);

int setSize(PSET\* s);

void \*findMax(PSET\* s);

void \*findMin(PSET\* s);

void deleteSet(PSET\* s);

void PrintAll(PSET\* s);

void PrintMenu();

#endif // !example.c

קובץ main:

#include "example.h"

//the function checks if the int data is equal

int isequalint(void\* element, void\* data)

{

if (\*(int\*)element == \*(int\*)data)

return 1;

return 0;

}

//the function checks if the char data is equal

int isequalst(void\* element, void\* data)

{

if (!(strcmp((char\*)element, (char\*)data)))

return 1;

return 0;

}

//the function checks if data (int) is greater than element

int GreaterInt(void\* element, void\* data)

{

if (\*(int\*)data > \*(int\*)element)

return 1;

return 0;

}

//the function checks if data (string) is greater than element

int GreaterStr(void\* element, void\* data)

{

if (strcmp((char\*)element, (char\*)data) < 0) //data is greater than element

return 1;

return 0;

}

//the function prints integers

void PrntNum(void\* data)

{

printf("%d ", \*(int\*)data);

}

//the function prints chars

void PrntChar(void\* data)

{

printf("%s ", (char\*)data);

}

//the function deletes the data indside the set

void DeleteInt(void\* data)

{

free(data);

}

//the function deletes the data indside the set

void DeleteChar(void\* data)

{

free(data);

}

//the function executes the int menu

PSET MenuInt(PSET s)

{

int choice;

int IntData;

int \*NumData;

printf("Enter an integer ('-999' to stop): ");

scanf("%d", &IntData);

while (IntData != -999)

{

NumData = (int\*)malloc(sizeof(int)); //creates a specific address for every variable thats being scanned from the user

if (NumData == NULL) //check the memory allocation

{

deleteSet(&s);

FreeSet(s);

Err\_Msg("Memory!");

}

\*NumData = IntData; //copying the data to the new address

createSet(NumData, &s); //creating the nodes and connecting them to the linked list

printf("Enter an integer ('-999' to stop): ");

scanf("%d", &IntData);

}

PrintMenu(); //shows the menu to the user

scanf("%d", &choice);

while (choice != 9) //the user chooses an option to execute (1-8 and 9 to end)

{

switch (choice)

{

case 1: //add new data

printf("Enter a new integer: ");

scanf("%d", &IntData);

NumData = (int\*)malloc(sizeof(int)); //creates a specific address for every variable thats being scanned from the user

if (NumData == NULL)

{

deleteSet(&s);

FreeSet(s);

Err\_Msg("Memory!");

}

\*NumData = IntData; //copying the data to the new address

addToSet(&s, NumData); //creats new node

break;

case 2: //delete data

printf("Enter an integer you want to delete from the set: ");

scanf("%d", &IntData);

if (deleteFromSet(&s, &IntData) == TRUE) //in case the data is in the set

printf("%d was removed from the set\n", IntData);

else //in case the data is not in the set

printf("%d wasn't found in the set!\n", IntData);

break;

case 3: //find data in the set

printf("Enter an integer you want to find the set: ");

scanf("%d", &IntData);

if (findInSet(&s, &IntData) == TRUE) //in case the data is in the set

printf("%d was found in the set\n", IntData);

else //in case the data is not in the set

printf("%d wasn't found in the set!\n", IntData);

break;

case 4: //check the size of the set

printf("There are %d integers in the set\n", setSize(&s));

break;

case 5: //find the maximum data in the set

if (findMax(&s) == NULL)

printf("There is no set!\n");

else printf("The maximum integer in the set is: %d\n", \*(int\*)findMax(&s));

break;

case 6: //find the minimum data in the set

if (findMin(&s) == NULL)

printf("There is no set!\n");

else printf("The minimum integer in the set is: %d\n", \*(int\*)findMin(&s));

break;

case 7: //delete the entire set

deleteSet(&s);

printf("The set has been deleted!\n");

break;

case 8: //print the entire set

PrintAll(&s);

break;

default: //in case the user didn't entered a valid option (1-9)

printf("Input Error!\n");

break;

}

PrintMenu();

scanf("%d", &choice);

}

return s;

}

//the function executes the char menu

PSET MenuChar(PSET s)

{

int choice;

char CharData[256];

char \*StringData;

printf("Enter a string ('end' to stop): ");

scanf("%s", CharData);

while (strcmp(CharData, "end"))

{

StringData = (char\*)malloc(sizeof(char)\*strlen(CharData) + 1); //creates a specific address for every variable thats being scanned from the user

if (StringData == NULL) //check the memory allocation

{

deleteSet(&s);

FreeSet(s);

Err\_Msg("Memory!");

}

strcpy(StringData, CharData); //copying the data to the new address

createSet(StringData, &s); //creating the nodes and connecting them to the linked list

printf("Enter a string ('end' to stop): ");

scanf("%s", CharData);

}

PrintMenu(); //shows the menu to the user

scanf("%d", &choice);

while (choice != 9) //the user chooses an option to execute (1-8 and 9 to end)

{

switch (choice)

{

case 1: //add new data

printf("Enter a new string: ");

scanf("%s", CharData);

StringData = (char\*)malloc(sizeof(char)\*strlen(CharData) + 1); //creates a specific address for every variable thats being scanned from the user

if (StringData == NULL)

{

deleteSet(&s);

FreeSet(s);

Err\_Msg("Memory!");

}

strcpy(StringData, CharData); //copying the data to the new address

addToSet(&s, StringData); //creats new node

break;

case 2: //delete data

printf("Enter a string you want to delete from the set: ");

scanf("%s", CharData);

if (deleteFromSet(&s, CharData) == TRUE) //in case the data is in the set

printf("%s was removed from the set\n", CharData);

else //in case the data is not in the set

printf("%s wasn't found in the set!\n", CharData);

break;

case 3: //find data in the set

printf("Enter a string you want to find the set: ");

scanf("%s", CharData);

if (findInSet(&s, CharData) == TRUE) //in case the data is in the set

printf("%s was found in the set\n", CharData);

else //in case the data is not in the set

printf("%s wasn't found in the set!\n", CharData);

break;

case 4: //check the size of the set

printf("There are %d strings in the set\n", setSize(&s));

break;

case 5: //find the maximum data in the set

if (findMax(&s) == NULL)

printf("The is no set!\n");

else printf("The maximum string in the set is: %s\n", (char\*)findMax(&s));

break;

case 6: //find the minimum data in the set

if (findMin(&s) == NULL)

printf("The is no set!\n");

printf("The minimum string in the set is: %s\n", (char\*)findMin(&s));

break;

case 7: //delete the entire set

deleteSet(&s);

printf("The set has been deleted!\n");

break;

case 8: //print the entire set

PrintAll(&s);

break;

default: //in case the user didn't entered a valid option (1-9)

printf("Input Error!\n");

break;

}

PrintMenu();

scanf("%d", &choice);

}

return s;

}

int main()

{

char ch;

PSET s;

while (1) //loop to choose between chars and integers

{

printf("Choose Integers or Chars (i for integers, c for chars): ");

scanf(" %c", &ch);

if ((ch == 'i') || (ch == 'c'))

break;

else

printf("Input Error!\n");

}

if (ch == 'i') //the program will work with integers

{

Init(&s, isequalint, GreaterInt, PrntNum, DeleteInt); //initalizing manager structure

s = MenuInt(s);

deleteSet(&s);

}

else //the program will work with strings

{

Init(&s, isequalst, GreaterStr, PrntChar, DeleteChar); //initalizing manager structure

s = MenuChar(s);

deleteSet(&s);

}

printf("End of program!\n");

FreeSet(s);

getch();

return 0;

}

קובץ implementation:

#include "example.h"

//structures definition

typedef struct SET

{

struct node \*head;

struct node \*tail;

int size; //the number of nodes in the set

//pointers to specific functions

int(\*isequal)(void\*, void\*);

int(\*isgreater)(void\*, void\*);

void(\*print)(void\*);

void(\*delete)(void\*);

}SET;

typedef struct node

{

void \*data;

struct node \*next;

struct node \*prev;

}node;

//the function frees set

void FreeSet(PSET s)

{

free(s);

}

//the function initializing the PSET at first

void Init(PSET\* s, int(\*isequal)(void\*, void\*), int(\*isgreater)(void\*, void\*), void(\*print)(void\*), void(\*delete)(void\*))

{

(\*s) = (PSET)malloc(sizeof(SET));

if ((\*s) == NULL)

Err\_Msg("Memory!");

(\*s)->head = NULL;

(\*s)->tail = NULL;

(\*s)->size = 0;

(\*s)->isequal = isequal;

(\*s)->isgreater = isgreater;

(\*s)->print = print;

(\*s)->delete = delete;

}

//the function creates a linked list

void createSet(void\* variable, PSET\* s)

{

node\* temp;

temp = (node\*)malloc(sizeof(node));

if (temp == NULL)

{

(\*s)->delete((\*s)->head);

FreeSet(s);

Err\_Msg("Memory!");

}

temp->prev = NULL;

temp->next = NULL;

temp->data = variable;

(\*s)->size++;

if ((\*s)->head == NULL)

{

(\*s)->head = temp;

(\*s)->tail = temp;

}

else

{

(\*s)->tail->next = temp;

temp->prev = (\*s)->tail;

(\*s)->tail = temp;

}

}

//the function closes the program in case not enough memory was allocated

void Err\_Msg(char\*s)

{

printf("%s", s);

exit(1);

}

//the function adds new element to and exsistant node

void addToSet(PSET\* s, void \*Element)

{

int i = 0;

node\* temp = (\*s)->head;

while((i < (\*s)->size) && (temp!=NULL))

{

if ((\*s)->isequal(Element, temp->data)) //checks if Element is in the set

{

printf("The data already exsits int the SET!\n");

break;

}

temp = temp->next;

i++;

}

if (i == (\*s)->size) //in case the data isn't already in the set

{

createSet(Element, s);

printf("The data has been added to the set!\n");

}

else if (temp == NULL) //in case the set is empty

{

createSet(Element, s);

printf("The data has been added to the set!\n");

}

}

//the function deletes a node from the linked list

BOOL deleteFromSet(PSET\* s, void\* Element)

{

int i;

node \*temp = (\*s)->head;

if (temp == NULL)

return FALSE;

for (i = 0; i < (\*s)->size; i++)

{

if ((\*s)->isequal(Element, temp->data)) //checks if Element is equal to temp->data

{

if (temp->prev == NULL)

{

(\*s)->head = (\*s)->head->next;

(\*s)->head->prev = NULL;

(\*s)->delete(temp->data);

free(temp);

}

else if (temp->next == NULL)

{

(\*s)->tail = (\*s)->tail->prev;

(\*s)->tail->next = NULL;

(\*s)->delete(temp->data);

free(temp);

}

else

{

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

(\*s)->delete(temp->data);

free(temp);

}

(\*s)->size--;

return TRUE;

}

temp = temp->next;

}

return FALSE;

}

//the function checks if the data appears in the set

BOOL findInSet(PSET\* s, void\* Element)

{

int i;

node\* temp = (\*s)->head;

if (temp == NULL)

return FALSE;

for (i = 0; i < (\*s)->size; i++)

{

if ((\*s)->isequal(Element, temp->data)) //checks if Element is equal to temp->data

return TRUE;

temp = temp->next;

}

return FALSE;

}

//the function checks how many nodes are in the linked list

int setSize(PSET\* s)

{

return (\*s)->size;

}

//the function checks which data is max

void \*findMax(PSET\* s)

{

int i;

void\* Max;

node\* temp = (\*s)->head;

if (temp == NULL)

{

Max = NULL;

return Max;

}

Max = temp->data;

for (i = 0; i < (\*s)->size; i++)

{

if (((\*s)->isgreater)(Max, temp->data)) //checks if Max is greater than temp->data

Max = temp->data;

temp = temp->next;

}

return Max;

}

//the function checks which data is min

void \*findMin(PSET\* s)

{

int i;

void\* Min;

node\* temp = (\*s)->head;

if (temp == NULL)

{

Min = NULL;

return Min;

}

Min = temp->data;

for (i = 0; i < (\*s)->size; i++)

{

if (!((\*s)->isgreater)(Min, temp->data)) //checks if Min is lower than temp->data

Min = temp->data;

temp = temp->next;

}

return Min;

}

//the function deletes the entire linked list

void deleteSet(PSET \*s)

{

node\* temp;

while ((\*s)->head != NULL)

{

temp = (\*s)->head;

(\*s)->head = (\*s)->head->next;

(\*s)->delete(temp->data); //delete node

free(temp);

(\*s)->size--;

}

}

//the function prints the entire set

void PrintAll(PSET \*s)

{

node\* temp = (\*s)->head;

if (temp == NULL)

printf("The set is empty!");

else while (temp != NULL)

{

(\*s)->print(temp->data); //prints the data

temp = temp->next;

}

printf("\n");

}

//the function prints the menu to the user

void PrintMenu()

{

printf("Choose a number from the following functions:\n");

printf("1. Add data to the set.\n");

printf("2. Delete data from the set.\n");

printf("3. Search for data in the set.\n");

printf("4. Check the size of the set.\n");

printf("5. Find the maximum data.\n");

printf("6. Find the minimum data.\n");

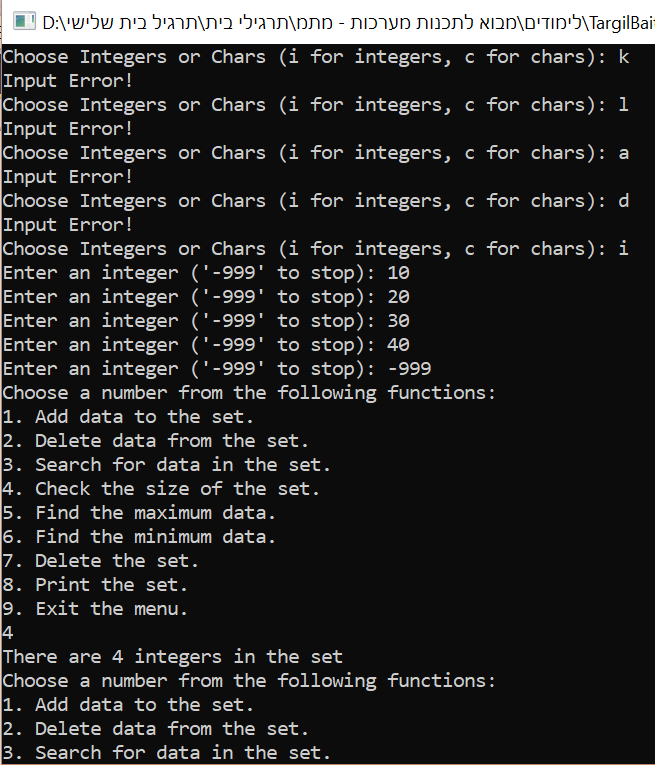
printf("7. Delete the set.\n");

printf("8. Print the set.\n");

printf("9. Exit the menu.\n");

}

פלט 1 לדוגמא:



פלט 2 לדוגמא:

