https://github.com/tonujet/c-embedded/blob/main/basics/L4/code/src/main.c

Вивід в консолі:

```
Max size: 255symbols
Save to stack:weqwewqewq
Store in heap:@#!!@weqeqwe!@#!@^%%
|-------------------------------------------------------------------------
| Stack: Uppercase using pointer: WEQWEWQEWQ
| Stack: Uppercase using variable in the stack: WEQWEWQEWQ
| -----------------------------------------------------------------------
| Heap: Uppercase using first pointer it: @#!!@WEQEQWE!@#!@^%%
| Heap: Uppercase using second pointer it: @#!!@WEQEQWE!@#!@^%%
| -----------------------------------------------------------------------
Exit?(Y/N)n

Max size: 255symbols
Save to stack:1123131
Store in heap:ghghgh
|-------------------------------------------------------------------------
| Stack: Uppercase using pointer: 1123131
| Stack: Uppercase using variable in the stack: 1123131
| -----------------------------------------------------------------------
| Heap: Uppercase using first pointer it: GHGHGH
| Heap: Uppercase using second pointer it: GHGHGH
| -----------------------------------------------------------------------
Exit?(Y/N)y
```

Код:

```c
#include "stringUtils.h"
#include "ui.h"
#include "malloc.h"



// Variant 3
// 3. Привести строку до великих символів
int main(int argc, char **argv) {
    char stringInStack[255] = {0};
    char *stringInHeap = malloc( Size: 255);
    while(1) {

        /* !!!!
         * char *stringInStack = "value";
         * Doesnt work properly, because we create pointer to stack memory and don`t
         * assign length(impossible to do). That means that our future changes will be
         * applied not to our variable, but to other information stored in thr stack at the current
         * moment. Such changes will ruin our program, and we will get error exit code;
         * It is possible to do, if we allocate memory in heap, and create our stringInStack there
        */

        inputVars(stringInStack, stringInHeap);
        char *upperInStack = toUpperCase( string: stringInStack);
        char *upperInHeap = toUpperCase( string: stringInHeap);
        printResults(stringInStack, upperInStack, stringInHeap, upperInHeap);
        if(exitUI() == 1) break;
    }
    free( Memory: stringInHeap);
}
```

- stringUtils.h

```c
#ifndef STRING_UTILS_H
#define STRING_UTILS_H

#define UPPER_LOVER_OFFSET 32

char* toUpperCase(char *string);
char toUpperChar(char c);

#endif
```

- stringUtils.c

```c
#include "stringUtils.h"

char *toUpperCase(char *string) {
    for (int i = 0;; i++) {
        char currChar = string[i];
        if (currChar == '\0') break;
        char upperChar = toUpperChar( c: currChar);
        if (currChar != upperChar) string[i] = upperChar;
    }
    return string;
}


char toUpperChar(char c) {
    if (c >= 97 && c <= 122) {
        return c - UPPER_LOVER_OFFSET;
    }
    return c;
}

```

- ui.h

```c
#ifndef UI_H
#define UI_H

void inputVars(char *stringInStack, char *stringInHeap);
void printResults(char *stringInStack, char *upperInStack, char *stringInHeap, char *upperInHeap);
int exitUI();
void flushInputBuffer();

#endif
```

- ui.c

```c
#include <stdio.h>
#include "stringUtils.h"
#include "ui.h"

void inputVars(char *stringInStack, char *stringInHeap){
    printf( format: "Max size: 255symbols\n");

    printf( format: "Save to stack:");

    scanf( format: "%255s", stringInStack);
    flushInputBuffer();

    printf( format: "Store in heap:");
    scanf( format: "%255s", stringInHeap);
    flushInputBuffer();
}

void printResults(char *stringInStack, char *upperInStack, char *stringInHeap, char *upperInHeap){
    printf( format: "|-------------------------------------------------------------------------------\n");
    printf( format: "| Stack: Uppercase using pointer: %s\n", stringInStack);
    printf( format: "| Stack: Uppercase using variable in the stack: %s\n", upperInStack);
    printf( format: "| -------------------------------------------------------------------------------\n");
    printf( format: "| Heap: Uppercase using first pointer it: %s\n",stringInHeap);
    printf( format: "| Heap: Uppercase using second pointer it: %s\n", upperInHeap);
    printf( format: "| -------------------------------------------------------------------------------\n");
}
```

```c
int exitUI(){
    printf( format: "Exit?(Y/N)");
    char c;
    scanf( format: "%c", &c);
    flushInputBuffer();

    c = toUpperChar(c);
    printf( format: "\n");
    if(c == 'Y') return 1;
    else if(c == 'N') return 0;
    else {
        printf( format: "Must be Y/N\n");
        return exitUI();
    }
}

void flushInputBuffer(){
    int temp;
    while ((temp = getchar()) != '\n' && temp != EOF);
}
```