

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт
з лабораторної роботи № 3
з дисципліни «Системне програмне забезпечення»
на тему: «Керування оперативною пам'яттю»

Виконав: ст. гр. КІ-302

Радевич-Винницький Я.А.

Перевірила: викладач

Ногаль М.В.

Мета роботи: Навчитися керувати оперативною пам'яттю в середовищі операційної системи, розробляти програми керування пам'яттю.

Варіант: 18.

Завдання:

1. Розробити програми для роботи з віртуальною пам'яттю. Для програмної реалізації використовувати середовище програмування Microsoft Visual Studio, мова програмування – C/C++, інтерфейс консольний.
2. Розробити програму, яка демонструє управління структурою даних типу «Дек» (черга з двома кінцями), елементами якого є значення типу short. Дек реалізувати за допомогою динамічного масиву розміром 7 Кб., пам'ять під який виділити за допомогою функції VirtualAlloc(). Операції, що виконуються над Деком: •перевірити Дек – порожній чи не порожній; •додати елемент в лівий кінець Дека; •додати елемент в правий кінець Дека; •видалити елемент ліворуч; •видалити елемент праворуч; •переглянути елемент ліворуч; •переглянути елемент праворуч; •вивести елементу Деку на екран.

Виконання завдання:

1. Створено програмний проєкт, який реалізує поставлене завдання.

Код файлу main.cpp:

Лістинг 1

```
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>
#define DEQUE_CAPACITY_Byte 7168
#define DEQUE_CAPACITY (DEQUE_CAPACITY_Byte / sizeof (short));
typedef
struct
{
    short* data;
    size_t size;
    size_t capacity;
} Deque;

void createDeque(Deque *deque)
{
    deque->data = (short*) VirtualAlloc(NULL, DEQUE_CAPACITY_Byte, MEM_COMMIT |
    MEM_RESERVE, PAGE_READWRITE);
    if (deque->data == NULL)
    {
        perror("Error allocating memory for deque");
        exit(EXIT_FAILURE);
    }
    deque->size = 0;
    deque->capacity = DEQUE_CAPACITY;
}

int isEmpty(Deque *deque)
{

```

```

        return deque->size == 0;
    }

    void addLeft(Deque* deque, short value) {
        if (deque->size < deque->capacity) {
            for (size_t i = deque->size; i > 0; --i) {
                deque->data[i] = deque->data[i - 1];
            }
            deque->data[0] = value;
            deque->size++;
        }
        else {
            fprintf(stderr, "Deque capacity exceeded\n");
            exit(EXIT_FAILURE);
        }
    }

    void addRight(Deque* deque, short value) {
        if (deque->size < deque->capacity) {
            deque->data[deque->size] = value;
            deque->size++;
        }
        else {
            fprintf(stderr, "Deque capacity exceeded\n");
            exit(EXIT_FAILURE);
        }
    }

    void removeLeft(Deque* deque) {
        if (!isEmpty(deque)) {
            for (size_t i = 0; i < deque->size - 1; ++i) {
                deque->data[i] = deque->data[i + 1];
            }
            deque->size--;
        }
        else {
            fprintf(stderr, "Deque is empty\n");
        }
    }

    void removeRight(Deque* deque) {
        if (!isEmpty(deque)) {
            deque->size--;
        }
        else {
            fprintf(stderr, "Deque is empty\n");
        }
    }

    short peekLeft(Deque* deque) {
        if (!isEmpty(deque)) {
            return deque->data[0];
        }
        else {
            fprintf(stderr, "Deque is empty\n");
            exit(EXIT_FAILURE);
        }
    }

    short peekRight(Deque* deque) {
        if (!isEmpty(deque)) {
            return deque->data[deque->size - 1];
        }
        else {
            fprintf(stderr, "Deque is empty\n");
        }
    }

```

```

        exit(EXIT_FAILURE);
    }
}

void printDeque(Deque* deque) {
    printf("Deque elements:\n");
    for (size_t i = 0; i < deque->size; ++i) {
        printf("%d ", deque->data[i]);
    }
    printf("\n");
}

void destroyDeque(Deque *deque)
{
    if (deque->data != NULL)
        VirtualFree(deque->data, 0, MEM_RELEASE);
}

int main()
{
    Deque deque;
    createDeque(&deque);

    printf("\nIs deque empty ? %s\n", isEmpty(&deque) ? "Yes" : "No");

    addLeft(&deque, 10);
    addRight(&deque, 20);
    addLeft(&deque, 5);
    addRight(&deque, 17);

    printf("\nIs deque empty ? %s\n", isEmpty(&deque) ? "Yes" : "No");
    printDeque(&deque);

    printf("\nPeek left: %d", peekLeft(&deque));
    printf("\nPeek right: %d\n", peekRight(&deque));

    removeLeft(&deque);
    removeRight(&deque);

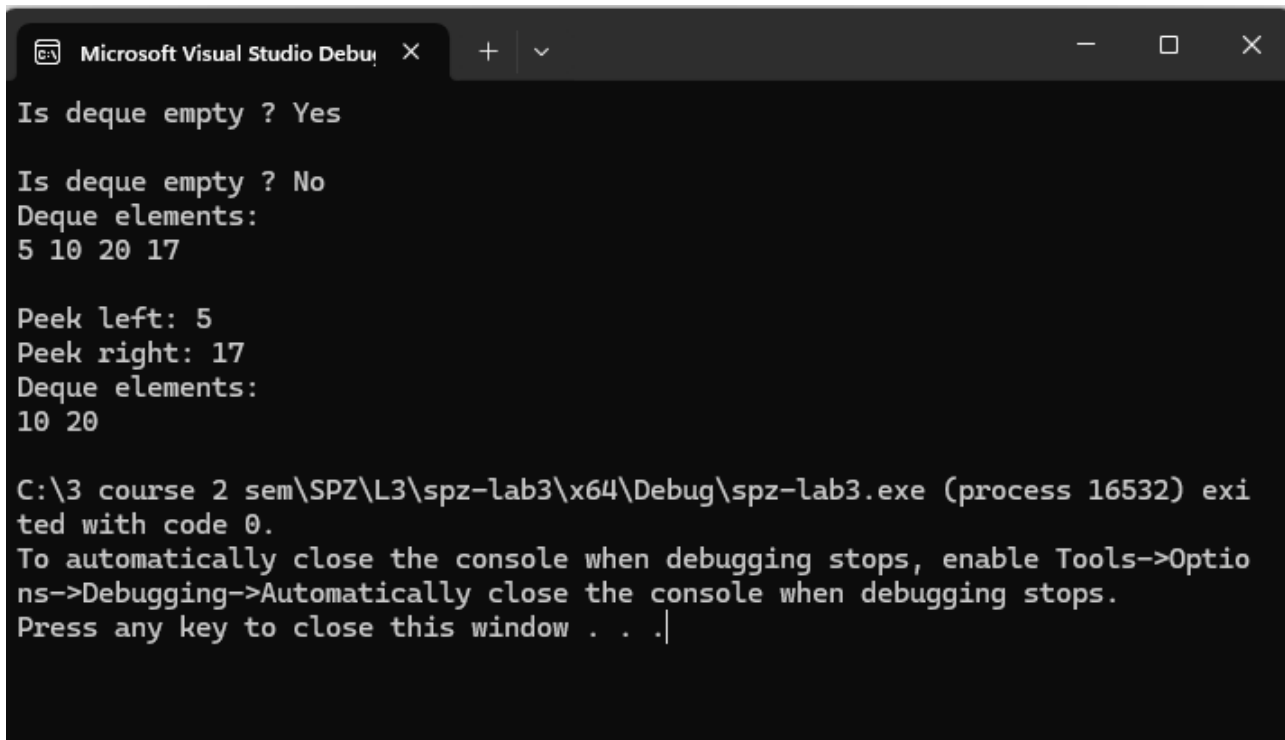
    printDeque(&deque);

    destroyDeque(&deque);

    return 0;
}

```

Результат роботи програми:



```
Microsoft Visual Studio Debug Console
Is deque empty ? Yes
Is deque empty ? No
Deque elements:
5 10 20 17

Peek left: 5
Peek right: 17
Deque elements:
10 20

C:\3 course 2 sem\SPZ\L3\spz-lab3\x64\Debug\spz-lab3.exe (process 16532) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Рис. 1 – робота програми

Висновок: під час виконання лабораторної роботи було вивчено основи керування оперативною пам'яттю в середовищі операційної системи та було розроблено програму, яка реалізовує структуру даних двобічної черги (deque).