Лабораторна робота №18

Романенко Дмитрий Кит 120д

Тема: Динамічні масиви.

Мета: Придбати практичні навички з використанням динамічних масивів.

Індивідуальне завдання

- розробити функцію, яка реалізує вставку в рядок “s” другий рядок “s2” в “i”-у позицію рядка “s”. Наприклад, `insert("abrakadabra", "TEXT2", 4)` повинна створити рядок “abraTEXT2kadabra”;
- розробити функцію видалення з рядка “s” усіх символів з індексами в заданому діапазоні. Наприклад, `reduce("abrakadabra", 4, 8)` повинна створити рядок “abrara” (без підрядка kadab).
- за допомогою функцій memcpy, memset створити функції додання та видалення елементів з динамічного масиву вашої прикладної області

Хід роботи

```
int main() {

    char mass[] = "babkakadavra";
    char mass2[] = "TEXT2";
    char* result = insert(mass, mass2, 4);
    reduce(result, 4, 8);

    return 0;
}

char* insert(char* mass, char* mass2, int index) { ... }

void reduce(char* mass, int indexStart, int indexEnd) { ... }
```

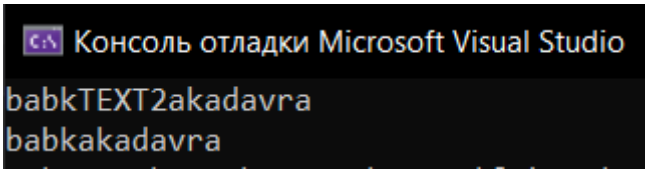
Рис.1 - Фрагмент коду

```
char* concatenation(char* mass, char* mass2, int index) {

    int size = strlen(mass) + strlen(mass2);
    char* s = (char*)malloc((size + 1) * sizeof(char));

    for (int i = 0; i < index; i++)
        s[i] = mass[i];
    for (int i = index; i < strlen(mass2) + index; i++)
        s[i] = mass2[i - index];
    for (int i = strlen(mass2) + index; i < size; i++)
        s[i] = mass[i - strlen(mass2)];

    s[size] = '\0';

    printf("%s\n", s);

    return s;
}
```

Рис.2 - функція вставки тексту.

```
void reduce(char* mass, int indexStart, int indexEnd) {

    int size = strlen(mass) - (indexEnd - indexStart + 1);
    char* s = (char*)malloc((size + 1) * sizeof(char));

    for (int i = 0; i < indexStart; i++)
        s[i] = mass[i];

    for (int i = indexEnd + 1; i < strlen(mass); i++)
        s[i - (indexEnd - indexStart + 1)] = mass[i];
    s[size] = '\0';
    printf("%s", s);
    return;
}
```

Рис.3 - функція видалення тексту.



Результат виконання програми

```
int addMessage(struct Message* messages, int size) {

    size++;
    Message* mass = (Message*)malloc(size * sizeof(Message));
    memcpy(mass, messages, sizeof(Message) * (size - 1));
    mass[size - 1] = create(size-1);
    memcpy(messages, mass, sizeof(Message) * size);

    return size;
}
```

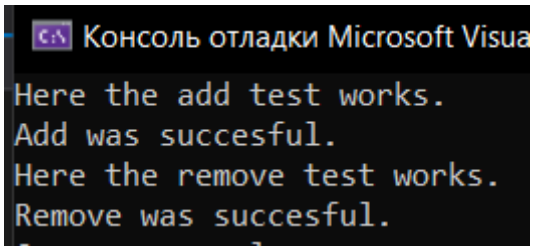
Рис.4 - функція додавання елементів.

```
int removeMessage(struct Message* messages, int size, int index) {

    size--;
    Message* mass = (Message*)malloc(size * sizeof(Message));
    for (int i = 0; i < size; i++)
    {
        if (i < index)
        {
            mass[i] = messages[i];
        }
        else
        {
            mass[i] = messages[i + 1];
        }
    }
    memcpy(messages, mass, sizeof(Message) * size);

    return size;
}
```

Рис.5 - функція видалення елемента з використанням memcpy.

```
void testAdd() {

    printf("Here the add test works.\n");

    int size = 2;
    Message* mass = (Message*)malloc(size * sizeof(Message));
    for (int i = 0; i < size; i++)
    {
        mass[i] = create(i);
    }
    size = addMessage(mass, size);

    if (mass[size - 1].encoding > 0 && size == 3)
    {
        printf("Add was succesful.\n\n");
    }
    else
    {
        printf("Add was not succesful.\n\n");
    }

}
```

Рис.5 - приклад тесту для додавання.



Результат тестування функцій наведених вище.

Висновок: в лабораторній роботі отримані навички з роботою динамічних рядків, а також навчились використовувати memset, memcpy для додавання та видалення елементів з динамічного масиву прикладної області Message.