МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №1
З предмету «Алгоритми та структури даних»

Виконав
Студент групи КН-36а
Рубан Ю.Д.
Перевірила:
ас.
Бородіна І. О.

Харків
2017

Лабораторна робота 1. БАЗОВІ СТРУКТУРИ ДАНИХ

Мета роботи: познайомитися з базовими структурами даних (список, черга, стек) та отримати навички програмування алгоритмів, що їх обробляють.

1.1. Теоретичний матеріал, необхідний для виконання лабораторної роботи 1

Стеки і черги – це динамічні множини, в яких елемент видаляється з множини операцією Delete, яка не задається довільно, а визначається структурою множини. Саме зі стеку (stack) можна видалити тільки той елемент, який був у нього доданий останнім: стек працює за принципом «останнім прийшов – першим пішов» (last-in, first-out – LIFO). З черги (queue), навпаки, можна видалити тільки той елемент, який знаходився в черзі довше за всіх: працює принцип «першим прийшов – першим пішов» (first-in, first-out – FIFO). У зв'язаному списку (або просто списку; linked list) елементи лінійно впорядковані, але порядок визначається не номерами, як у масиві, а вказівниками, що входять до складу елементів списку. Списки є зручним способом реалізації динамічних множин. Якщо кожен, хто стоїть у черзі, запам'ятає, хто за ним стоїть, після чого всі безладно розсядуться, то вийде однобічно зв'язаний список; якщо він запам'ятає ще й того, хто попереду, буде двобічно зв'язаний список. Іншими словами, елемент двобічно зв'язаного списку (doubly linked list) – це запис, що містить три поля: key (ключ) і два вказівники next (наступний) і prev (попередній). Крім цього, елементи списку можуть містити додаткові дані. У різних ситуаціях використовуються різні види списків. В односторонньому незв'язаному (singly linked) списку відсутні поля prev. У впорядкованому (sorted) списку елементи розташовані в порядку зростання ключів таким чином, що у голови списку ключ найменший, а у хвоста списку – найбільший, на відміну від неврегульованого (unsorted) списку. У кільцевому списку (circular list) поле prev голови списку вказує на хвіст списку, а поле next хвоста списку вказує на голову списку.

1.2. Завдання до лабораторної роботи 1

Розробити програму, яка читає з клавіатури послідовність N цілих чисел (1<N<256), жодне з яких не повторюється, зберігає їх до структури даних (згідно з завданням) та видає на екран такі характеристики:

- кількість елементів;

- середнє арифметичне збережених елементів;

- мінімальний та максимальний елемент;

- четвертий елемент послідовності;

- елемент, що йде перед мінімальним елементом.

Наголосимо, що всі характеристики потрібно визначити із заповненої структури даних. Дозволено використовувати лише ті операції, що притаманні заданій структурі, наприклад, заборонено отримувати доступ до елемента із довільною позицією у черзі, яку реалізовано на базі масиву. Використовувати готові реалізації структур даних (наприклад, STL) заборонено.

1.3. Варіанти завдань до лабораторної роботи 1

Для виконання завдання до лабораторної роботи потрібно використати такі структури даних:

Варіант 1: черга.

Варіант 2: стек.

Варіант 3: однобічно зв’язаний список.

Варіант 4: двобічно зв’язаний список.

Варіант 5: кільцевий список.(Виконаний варіант)

Варіант 6: масив із довільним доступом.

Хід виконання роботи

На основі теоретичного матеріалу лабороторної роботи №1 було створено динамічну структуру даних – кільцевий зв’язний список, який працює так само, як і однобічний зв’язний список але останній єлемент вказує не на NIL, а на перший єлемент. Також за допомогою цього списку було виконано завдання лабораторної роботи.

Код реалізації:

```
#include<iostream>
using namespace std;

class cList
{
public:
	void add(int val)
	{
		cList *temp = new cList;
		temp->data = val;
		temp->next = head;
		if (head == 0)
		{
			head = tail = temp;
		}
		else
		{
			tail->next = temp;
			tail = temp;
		}
		count++;
	}
	int getCount()const { return count; }
	int getElemViaIndex(int index)
	{
		cList *temp = this->head;
		int c = this->getCount();
		if (index < 0)
		{
			for (int i = 0; i < c + index; i++)
			{
				temp = temp->next;
			}
		}
		else
		{
			for (int i = 0; i < index; i++)
			{
				temp = temp->next;
			}
		}
		return temp->data;
	}
	int getMax()
	{
		cList temp = *this;
		int max = temp.getElemViaIndex(0);
		for (int i = 0; i < temp.count; i++)
		{
			if (max < temp.getElemViaIndex(i))
			{
				max = temp.getElemViaIndex(i);
```

```
                }
            }
            return max;
        }
        int getMin()
        {
            cList temp = *this;
            int min = temp.getElemViaIndex(0);
            for (int i = 0; i < temp.count; i++)
            {
                if (min > temp.getElemViaIndex(i))
                {
                    min = temp.getElemViaIndex(i);
                }
            }
            return min;
        }
        int getPreMinElem()
        {
            cList temp = *this;
            int a = temp.getMin();
            int index = 0;
            for (int i = 0; i < temp.count; i++)
            {
                if (a == temp.getElemViaIndex(i))
                {
                    index = i;
                }
            }
            return temp.getElemViaIndex(index - 1);
        }

    private:
        cList *head;
        cList *next;
        cList *tail;
        int data;
        int count = 0;
};
int main()
{
        cList c;
        c.add(2);
        c.add(3);
        c.add(7);
        c.add(16);
        c.add(20);
        int avg = 0;
        for (int i = 0; i < c.getCount(); i++)
        {
            avg += c.getElemViaIndex(i);
        }
        avg = avg / c.getCount();
        cout << "avg = " << avg << endl;
        int min = c.getMin();
        cout << "min = " << min << endl;
        int max = c.getMax();
        cout << "max = " << max << endl;
        int fourthElem = c.getElemViaIndex(3);
        cout << "4 elem = " << fourthElem << endl;
        cout << "preMin elem = " << c.getPreMinElem() << endl;
        system("pause");
        return 0;
}
```

Висновок

У даній лабораторній роботі я вивчив базові структури даних, навчився їх реалізовувати на прикладі кільцевого зв’язного списку, і закріпив знання з алгоритму їх обробки.