*Елгин Илья, ИУ7-34*

*Типы и структуры данных*

*Лабораторная работа 4*

*Работа со стеком*

*Вариант 5*

1. **Техническое задание**
   1. **Задача:**

Ввести целые числа в 2 стека. Используя третий стек отсортировать все введенные данные. Реализовать стек: а)массивом; б) списком. Все стандартные операции со стеком должны быть оформлены подпрограммами. При реализации стека списком в вывод текущего состояния стека добавить просмотр адресов элементов стека и создать свой список или массив свободных областей (адресов освобождаемых элементов) с выводом его на экран.

* 1. **Входные данные**

Числа (элементы стека),

Номер выбранной операции:

1-Ввести и отсортировать стек в виде списка

2-Ввести и отсортировать стек в виде массива

3-Добавит в стек элемент

4-Удалить последний элемент из стека

5-Вывести стек

6-Вывести адреса

7-Изменить реализацию стека

0-Выход

* 1. **Выходные данные**

Элементы стека,

адреса элементов стека в виде списка.

* 1. **Аварийные ситуации**

В случае переполнения стека, или если пользователь пытается удалить элемент из пустого стека, или если пользователь введёт некорректные данные, программа выдаст сообщение об ошибке.

* 1. **Способ обращения программой**

Исполняемый файл app.exe. Запускается из командной строки, аргументы строки не требуются.

1. **Структура данных**

В программе было использовано 2 реализации стека - массив и список. Содержимое стека - адреса памяти (стека). Структура стека для массива содержит в себе массив элементов, указатель на вершину массива и максимальное количество элементов. Структура стека для списка содержит в себе значение элемента (число), а также указатель на следующий элемент (next).

*Реализация массивом:*

**typedef struct array**

**{**

**int n; -** *количество элементов*

**int elements[MAX\_SIZE \* 2 + 1];*-***  *массив*

**}array;**

*Реализация списком:*

**typedef struct list**

**{**

**int val; - з***начение элемента*

**struct list \*next; -***ссылка на следующий элемент*

**}list;**

**Алгоритм**

Вывод пунктов меню.

Пока пользователь не выбрал 0:

1:

Запросить у пользователя 2 стека в виде списка

Отсортировать их в один стек

Вывести стек

2:

Запросить у пользователя 2 стека в виде массива

Отсортировать их в один стек

Вывести стек

3:

Если стек не переполнен и проинициализирован

Запросить у пользователя элемент

Push() элемент в стек

4:

Если стек не пустой и проинициализирован

Pop() элемент из стека

5:

Если стек не пустой и проинициализирован

Вывести стек

6:

Если стек представлен в виде списка

Вывести адреса

7:

Если стек представлен в виде списка

Переложить элементы из списка в массив

Иначе Если стек представлен в виде массива

Переложить элементы из массива в список

Алгоритм сортировки:

Пока один из стеков не пуст

Если второй стек пут или в первом лежит больший элемент чем во втором

Вытащить элемент из первого стека

Пока в сортировочном стеке крайний элемент не будет больше вытащенного или сортировочный стек не будет пуст

Достать элемент из сортировочного стека и положить в первый стек

Положить элемент в сортировочный стек

Иначе

Вытащить элемент из второго стека

Пока в сортировочном стеке крайний элемент не будет больше вытащенного или сортировочный стек не будет пуст

Достать элемент из сортировочного стека и положить во второй стек

Положить элемент в сортировочный стек

* 1. **Тесты**

|  |  |  |
| --- | --- | --- |
| **Опция меню** | **Значение\случай** | **Сообщение** |
| **3-Добавит в стек элемент** | Добавление 101 элемента | В стеке максимальное количество элементов, добавление невозможно. |
| **4-Удалить последний элемент из стека** | Попытка удалить элемент из пустого стека. | Стек пуст. |
| **5-Вывести стек** | Пустой стек | Стек пуст. |
| **5-Вывести стек** | Вывод стека | 0 1 1 2 5 6 23 31 46 78 79 180 |
| **1-Ввести и отсортировать стек в виде списка**  **аааааааааааа** | Ввод букв вместо цифр | Некорректный ввод, повторите. |

1. **Заключение**

В результате работы были реализованы функции работы со стеком, а также были сделаны обе реализации стека - через массив и через список. Массив дает преимущество по скорости, список даёт преимущество по памяти. Массив использовать целесообразно при известном количестве элементов.

При работе программы фрагментация памяти почти не происходит. Список лучше использовать при неопределённом количестве элементов в стеке, массив же наоборот при фиксированном количестве элементов.

Для данной программы список эффективнее массива по памяти при заполненности стека менее 50 элементов при максимальном числе элементов 100, по памяти же всегда выгоднее массив.

|  |  |  |
| --- | --- | --- |
|  | **push**  **(добавление)** | **pop**  **(удаление)** |
| **Стек массивом** | 208 тактов | 64 тактоа |
| **Стек списком** | 6736 тактов | 3568 тактов |

1. **Ответы на вопросы**

**1) Что такое стек?**

Стек – это последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины. Стек функционирует по принципу: последним пришел – первым ушел, Last In – First Out

(LIFO).

**2) Каким образом и сколько памяти выделяется под хранение стека при различной его реализации?**

Массив: Включает количество элементов и сами элемент, выделяет один сегмент памяти под все элементы, массив выделяет память под максимальное количество элементов и по памяти эффективен только при полном заполнении стека.

Список: Включает элементы и адреса на них, тоесть занимает в 2 раза больше памяти чем массив при полном заполнении стека, при заполнении стека меньше половины эффективнее по памяти чем массив, выделяет независимые сегменты памяти под каждый элемент.

**3) Каким образом освобождается память при удалении элемента стека при различной**

**реализации стека?**

Массив: количество элементов уменьшается на 1 , память не освобождается.

Список: память под данный элемент освобождается.

**4) Что происходит с элементами стека при его просмотре?**

Элемент извлекается с вершины стека и выводится на экран после чего данная операция повторяется со следующим элементом.

**5) Каким образом эффективнее реализовывать стек? От чего это зависит?**

Для эффективности по скорости лучше использовать массив так как ему не нужно освобождать и резервировать память под элементы, для эффективности по памяти лучше пользоваться списком , так как память выделяемая под список зависит от количества элементов в нём.

Для данной программы список эффективнее массива по памяти при заполненности стека менее 50 элементов при максимальном числе элементов 100, по памяти же всегда выгоднее массив.