Типы и структуры данных
Лабораторная работа 4
Вариант 1
Работа со стеком

Сиденко Анастасия

ИУ7-33Б

Техническое задание:

Задача:

Реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) и в виде односвязного линейного списка; оценить преимущества и недостатки каждой реализации: получить представление о механизмах выделения и освобождения памяти при работе со стеком.

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывод текущего состояния стека. 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.

Распечатайте убывающие серии последовательности целых чисел в обратном порядке.

Входные данные: числа.

Выходные данные: стек, в котором заменены все убывающие последовательности на обратные.

Взаимодействие с программой

Взаимодействие через консольное меню.

Вводятся элементы стека по одному, можно удалить или вывести стек.

```
Выберите, что хотите сделать?
1. Добавить элемент
2. Удалить элемент
3. Распечатать стек
4. Выйти из программы
```

Аварийные ситуации:

1. Некорректные данные. Вывод сообщения.

2. Стек переполнен. Вывод сообщения.

3. Стек пуст. Вывод сообщения.

Используемые структуры:

Структура односвязный список, где стек хранится.

```
struct stack {
    int data;
    struct stack *next;
};
```

Структура массив, где стек хранится.

```
int *st;
int kol = 0;
```

Массив свободных областей памяти.

```
int *massiv_free;
int free_kol = 0;
```

Алгоритмы:

1. Вывод меню и возвращение значения номера алгоритма, который нужно сделать

```
int menu();
```

2. Добавление элемента в стек

```
struct stack *stack_push(struct stack *st, int data);
```

3. Удаление элемента из стека

struct stack *stack_pop(struct stack *st, int *dst);

4. Печать элементов стека с заменой убывающих последовательностей на обратные

void stack_print_list(struct stack **st);

void stack_print_massiv(int *st, int kol);

Тесты:

1. Некорректные данные

Попытка удаления из пустого стека

```
Как будем хранить стек?
1. В массиве
2. В списке
2
Выберите, что хотите сделать?
1. Добавить элемент
2. Удалить элемент
3. Распечатать стек
4. Выйти из программы
2
Стек пустой
```

Попытка печати пустого стека

```
Как будем хранить стек?
1. В массиве
2. В списке
2
Выберите, что хотите сделать?
1. Добавить элемент
2. Удалить элемент
3. Распечатать стек
4. Выйти из программы
3
Стек пустой
```

Ввод некорректных данных

```
Как будем хранить стек?
1. В массиве
2. В списке
йц
Некорректный ввод. Попробуйте еще раз.
1
Выберите, что хотите сделать?
1. Добавить элемент
2. Удалить элемент
3. Распечатать стек
4. Выйти из программы
ке
Некорректный ввод. Попробуйте еще раз.
```

2. Правильные тесты

```

Выберите, что хотите сделать?
1. Добавить элемент
2. Удалить элемент
3. Распечатать стек
4. Выйти из программы
3
7 6 4 3 2
9 8 7 6 5 4 3 2
Время выполнения для матрицы в виде списка 40
```

```
Выберите, что хотите сделать?
1. Добавить элемент
2. Удалить элемент
3. Распечатать стек
4. Выйти из программы
3
Стек
7 6 4 3 2
9 8 7 6 5 4 3 2
Время выполнения для матрицы в виде массива 27
```

```
1
Введите элемент
45
Элемент добавлен в 0x7fdcf7700020
Свободные области
0x7fdcf7500020 0x7fdcf74006a0
```

```
1
Введите элемент
1
Элемент добавлен в 0x7fdcf7500020
Свободные области
0x7fdcf74006a0
```

Сравнение времени работы и используемой памяти.

При использовании массива память выделяется заранее вначале программы, при реализации списком память выделяется при заполнении стека.

Поэтому если количество элементов заранее неизвестно, массив будет выделен на большое число элементов, будет проигрывать по памяти.

А при заранее известном количестве на 67% использование массива (4 байта на элемент) эффективнее по памяти чем списка (16 байт).

И на 30% эффективнее по времени.

Вопросы:

1. Что такое стек?

Стек - последовательный односвязный список с переменной длиной, включение и исключение из которого происходят с одной стороны - его вершины.

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

Если стек реализован в виде статического или динамического массива (вектора), то для его хранения обычно отводится непрерывная область памяти ограниченного размера.

При включении элемента в стек сначала происходит выделение области памяти. В этом случае объем стека ограничивается только объемом доступной оперативной памяти.

3. Каким образом освобождается память при удалении элемента стека при различной реализации стека?

При реализации массивом память освобождается после завершения работы со стеком.

При реализации списком память очищается после каждого удаленного элемента.

4. Что происходит с элементами стека при его просмотре?

Классическая реализация стека предполагает, что просмотреть содержимое стека без извлечения (удаления) его элементов невозможно.

5. Каким образом эффективнее реализовывать стек? Отчего это зависит?

Эффективнее использовать массив, если заранее известно количество элементов стека и требуется более быстрая работа программы.

Список использовать эффективнее, если количество элементов массива неизвестно.

Вывод:

Использование стека в виде массива эффективнее чем стека в виде списка и по памяти, и по времени, если размер известен заранее. Так как в списке кроме элементов хранится еще и указатель.

Если размер стека неизвестен, то эффективнее по памяти использовать стек.