Отчет по Лабораторной работе № 25-26

по курсу: 1 фундаментальная информатика

студент группы : М8О-105Б-21 Титеев Рамиль Маратович , № по списку: 23

Адреса www, e-mail, jabber, skype: derol.gym@gmail.com

Работа выполнена: "21 мая 2022г"

Преподаватель: каф. 806 В.К.Титов

Входной контроль знаний с оценкой: _____

Отчет сдан "___"_____20___г., итоговая оценка _____

Подпись преподавателя _____

1. Тема: Абстрактные типы данных. Рекурсия.Модульное программирование на языке Си.
2. Цель работы: Автоматизация сборки программ модульной структуры на языке Си с Процедура: поиск и удаление максимального элемента.
3. Задание: АТД: Стек, Метод: сортировка линейным выбором, Процедура: поиск и удаление максимального элемента.
4. Оборудование(*лабораторное*):
ЭВМ _____, процессор _____, имя узла сети ____ с ОП ___ ГБ
НМД _____ ГБ. Терминал _____ адрес _____. Принтер _____
Другие устройства _____
Оборудование ПЭВМ студента, если использовалось:
Процессор Ryzen 4600 @ 6x 3.0 GHz , ОП 16384 МБ, НМД _____ГБ. Монитор Встроенный
Другие устройства _____

5. Программное обеспечение(*лабораторное*):
Операционная система семейства UNIX, наименование _____ версия _____
Интерпретатор команд: _____ версия _____
Система программирования: _____ версия _____
Редактор текстов: _____ версия _____
Утилиты операционной системы: _____
Прикладные системы и программы: _____
Местонахождение и имена файлов и программ данных: _____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства UNIX, наименование Ubuntu версия 20.04
Интерпретатор команд: bash версия _____
Система программирования: C версия _____
Редактор текстов: Emacs версия _____
Утилиты операционной системы: _____
Прикладные системы и программы: _____
Местонахождение и имена файлов и программ данных: /usr/bin , а также /bin

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

Описание метода:

Сначала находим и удаляем максимальный элемент из стека и запоминаем его. Затем создается временный стек. В него перекладываются все элементы стека. После этого в пустой исходный стек мы помещаем максимальный элемент, который был получен ранее. А затем из временного стека перекладываем элементы в основной. Это повторяется n(размер стека) раз. При этом глубина поиска максимального элемента уменьшается на 1 каждую итерацию.

Описание Процедуры:

Создаем временный стек и перекладываем в него все элементы исходного стека, находя при этом максимальный элемент. Затем перекладываем элементы из временного в основной стек. Если встретился максимальный элемент, то удаляем его, и перекладываем все оставшиеся элементы в исходный стек.

7. Сценарий выполнения работы [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

lab_25-26.cpp:

```
#include <stdio.h>
#include <stdlib.h>
#include <random>
#include "stack1.h"

void Sort(Stack &S){
    for(int i = Size(S); i>0; i--){
        int q = Del_max(S, i);
        Stack tmp;
        Init(tmp);
        while (!Empty(S)){
            Push(tmp, Pop(S));
        }
        Push(S, q);
        while (!Empty(tmp)){
            Push(S, Pop(tmp));
        }
    }
}

int main()
{
    srand(time(0));
    Stack S;
    Init(S); Tvalue v; int k;
    while(1){
        printf("-----------Menu----------\n");
        printf("| 1) Make random stack  |\n");
        printf("|    2) Print stack     |\n");
        printf("|     3) Push value     |\n");
        printf("|     4) Pop value      |\n");
        printf("|   5) Read top value   |\n");
        printf("|   6) Size of stack    |\n");
        printf("|  7) Sorting of stack  |\n");
        printf("|     8) Print Menu     |\n");
        printf("|        9) Exit        |\n");
        printf("-------------------------\n");
        printf("Choose an action ===> ");
        scanf("%d", &k);
```

```
        switch (k){
            case 1:{
                int s;
                printf("\nType length of stack (<100): ");
                scanf("%d",&s);
                printf("\n");
                while(s--){
                    Push(S,rand()%100);
                }
                break;
            }
            case 2:{
                printf("\n");
                Display(S);
                printf("\n");
                break;
            }
            case 3:{
                Tvalue s;
                printf("\n");
                printf("Type value: ");
                scanf("%d",&s);
                printf("\n");
                Push(S,s);
                break;
            }
            case 4:{
                Tvalue q = Pop(S);
                printf("\n%d was popped!\n\n", q);
                break;
            }
            case 5:{
                printf("\nTop value = %d\n\n", Top(S));
                break;
            }
            case 6:{
                printf("\nSize of stack = %d\n\n", Size(S));
                break;
            }
            case 7:{
                Sort(S);
                break;
            }
            case 8:{
                break;
            }
            case 9: return 0;
        }
    }
    return 0;
}
```

stack1.h:

```
// Реализация на массивах
```

```
#define N 100
#define Tvalue int

struct Stack{
    int first;
    Tvalue body[N];
};

void Init(Stack &S){
    S.first = 0;
}

int Empty(Stack S){
    return S.first == 0;
}

void Push(Stack &S, Tvalue V){
    if (S.first == N){
        printf("Stack is overflow!!!");
    }
    else{
        S.body[S.first++] = V;
    }
}

Tvalue Pop(Stack &S){
    if (!Empty(S)){
        return S.body[--S.first];
    }
    else{
        printf("Stack is Empty!!!");
        return 0;
    }
}

Tvalue Top(Stack &S){
    if (!Empty(S)){
        return S.body[S.first - 1];
    }
    else{
        printf("Stack is Empty!!!");
        return 0;
    }
}

int Size(Stack S){
    return S.first;
}

void Display(Stack S){
    printf("[ ");
    for (int i = 0; i < S.first; i++){
```

```
        printf("%d ", S.body[i]);
    }
    printf("]\n");
}

Tvalue Del_max(Stack &S, int n){
    Tvalue mx = 0;
    Stack tmp;
    Init(tmp);
    for (int i = 0; i<n; i++){
        if (Top(S)>mx) mx = Top(S);
        Push(tmp, Pop(S));
    }
    while (!Empty(tmp) && Top(tmp)!=mx){
        Push(S,Pop(tmp));
    }
    if(Top(tmp)==mx){
        Pop(tmp);
    }
    while (!Empty(tmp)){
        Push(S,Pop(tmp));
    }

    return mx;
}
```

stack2.h:

```
// Реализация на динамических структурах

#define N 100
#define Tvalue int

struct St{
    Tvalue Value;
    St *next;
};

struct Stack{
    St *first;
    int size;
};

void Init(Stack &S){
    S.first=0;
    S.size=0;
}

int Empty(Stack S){
    return S.first==0;
}

void Push(Stack &S, Tvalue V){
    if (S.size == N){
        printf("\nStack is overflow!!!\n");
```

```
    }
    else{
        St *t = new St;
        t->next = S.first;
        S.first=t;
        S.first->Value = V;
        S.size++;
    }
}

Tvalue Pop(Stack &S){
    if(!Empty(S)){
        Tvalue V = S.first->Value;
        St *elem = S.first;
        S.first = S.first->next;
        delete elem;
        S.size--;
        return V;
    }
    else{
        printf("\nStack is Empty!!!\n");
        return 0;
    }
}

Tvalue Top(Stack &S){
    if(!Empty(S)){
        return S.first->Value;
    }
    else{
        printf("\nStack is Empty!!!\n");
        return 0;
    }
}

int Size(Stack S){
    return S.size;
}

void Display(Stack S){
    if(Empty(S)){
        printf("\nStack is Empty!!!\n");
    }
    else{
        printf("[ ");
        St *t = S.first;
        while(t) {
            printf("%d ",t->Value);
            t=t->next;
        }
            printf("]\n");
    }
}
```

```
Tvalue Del_max(Stack &S, int n){
    Tvalue mx = 0;
    Stack tmp;
    Init(tmp);
    for (int i = 0; i<n; i++){
        if (Top(S)>mx) mx = Top(S);
        Push(tmp, Pop(S));
    }
    while (!Empty(tmp) && Top(tmp)!=mx){
        Push(S,Pop(tmp));
    }
    if(Top(tmp)==mx){
        Pop(tmp);
    }
    while (!Empty(tmp)){
        Push(S,Pop(tmp));
    }

    return mx;
}
```

Пункты 1-7 отчёта составляются ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя _____

8. Распечатка протокола (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем).

```
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ cat header.txt
*****************************************************************
*                   Лабораторная работа №25-26                  *
*              Абстрактные типы данных. Рекурсия.               *
*            Модульное программирование на языке Си.            *
*                Выполнил студент гр. М8О-105-Б                 *
*                    Титеев Рамиль Маратович                    *
*****************************************************************
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ cat lab_25-26.cpp
#include <stdio.h>
#include <stdlib.h>
#include <random>
#include "stack1.h"

void Sort(Stack &S){
    for(int i = Size(S); i>0; i--){
        int q = Del_max(S, i);
        Stack tmp;
        Init(tmp);
        while (!Empty(S)){
            Push(tmp, Pop(S));
        }
        Push(S, q);
        while (!Empty(tmp)){
            Push(S, Pop(tmp));
        }
    }
}

int main()
```

```
{
    srand(time(0));
    Stack S;
    Init(S); Tvalue v; int k;
    while(1){
        printf("-----------Menu----------\n");
        printf("| 1) Make random stack  |\n");
        printf("|    2) Print stack     |\n");
        printf("|     3) Push value     |\n");
        printf("|     4) Pop value      |\n");
        printf("|   5) Read top value   |\n");
        printf("|   6) Size of stack    |\n");
        printf("|  7) Sorting of stack  |\n");
        printf("|     8) Print Menu     |\n");
        printf("|        9) Exit        |\n");
        printf("-------------------------\n");
        printf("Choose an action ===> ");
        scanf("%d", &k);
        switch (k){
            case 1:{
                int s;
                printf("\nType length of stack (<100): ");
                scanf("%d",&s);
                printf("\n");
                while(s--){
                    Push(S,rand()%100);
                }
                break;
            }
            case 2:{
                printf("\n");
                Display(S);
                printf("\n");
                break;
            }
            case 3:{
                Tvalue s;
                printf("\n");
                printf("Type value: ");
                scanf("%d",&s);
                printf("\n");
                Push(S,s);
                break;
            }
            case 4:{
                Tvalue q = Pop(S);
                printf("\n%d was popped!\n\n", q);
                break;
            }
            case 5:{
                printf("\nTop value = %d\n\n", Top(S));
                break;
            }
            case 6:{
                printf("\nSize of stack = %d\n\n", Size(S));
                break;
            }
            case 7:{
                Sort(S);
                break;
            }
            case 8:{
                break;
```

```
            }
            case 9: return 0;
        }
    }
    return 0;
}
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ cat Makefile
stack_on_mass.exe: stack_on_mass.o
        g++ -c -o stack.o lab_25-26.cpp
        g++ -o stack.exe stack.o
stack_on_mass.o: stack1.h
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ make
g++ -c -o stack.o lab_25-26.cpp
g++ -o stack.exe stack.o
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ ./stack.exe
-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 1

Type length of stack (<100): 10

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 75 74 92 49 81 17 60 84 62 98 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 5

Top value = 98

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
```

```
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 4

98 was popped!

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 75 74 92 49 81 17 60 84 62 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 6

Size of stack = 9

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 3

Type value: 27

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
```

```
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 75 74 92 49 81 17 60 84 62 27 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 7
-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 92 84 81 75 74 62 60 49 27 17 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 3

Type value: 70

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 3
```

```
Type value: 54

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 92 84 81 75 74 62 60 49 27 17 70 54 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 6

Size of stack = 12

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 7
-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 92 84 81 75 74 70 62 60 54 49 27 17 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
```

```
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 9
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ cat lab_25-26.cpp
#include <stdio.h>
#include <stdlib.h>
#include <random>
#include "stack2.h"

void Sort(Stack &S){
    for(int i = Size(S); i>0; i--){
        int q = Del_max(S, i);
        Stack tmp;
        Init(tmp);
        while (!Empty(S)){
            Push(tmp, Pop(S));
        }
        Push(S, q);
        while (!Empty(tmp)){
            Push(S, Pop(tmp));
        }
    }
}

int main()
{
    srand(time(0));
    Stack S;
    Init(S); Tvalue v; int k;
    while(1){
        printf("-----------Menu----------\n");
        printf("| 1) Make random stack  |\n");
        printf("|    2) Print stack     |\n");
        printf("|     3) Push value     |\n");
        printf("|     4) Pop value      |\n");
        printf("|   5) Read top value   |\n");
        printf("|   6) Size of stack    |\n");
        printf("|  7) Sorting of stack  |\n");
        printf("|     8) Print Menu     |\n");
        printf("|        9) Exit        |\n");
        printf("-------------------------\n");
        printf("Choose an action ===> ");
        scanf("%d", &k);
        switch (k){
            case 1:{
                int s;
                printf("\nType length of stack (<100): ");
                scanf("%d",&s);
                printf("\n");
                while(s--){
                    Push(S,rand()%100);
                }
                break;
            }
            case 2:{
                printf("\n");
                Display(S);
```

```
                printf("\n");
                break;
            }
            case 3:{
                Tvalue s;
                printf("\n");
                printf("Type value: ");
                scanf("%d",&s);
                printf("\n");
                Push(S,s);
                break;
            }
            case 4:{
                Tvalue q = Pop(S);
                printf("\n%d was popped!\n\n", q);
                break;
            }
            case 5:{
                printf("\nTop value = %d\n\n", Top(S));
                break;
            }
            case 6:{
                printf("\nSize of stack = %d\n\n", Size(S));
                break;
            }
            case 7:{
                Sort(S);
                break;
            }
            case 8:{
                break;
            }
            case 9: return 0;
        }
    }
    return 0;
}
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ cat Makefile
stack_on_list.exe: stack_on_list.o
        g++ -c -o stack.o lab_25-26.cpp
        g++ -o stack.exe stack.o
stack_on_list.o: stack2.h
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ make
g++ -c -o stack.o lab_25-26.cpp
g++ -o stack.exe stack.o
constantfear@constantfear:~/projects/laboratory/2_semester/lab_25-26$ ./stack.exe
-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
------------------------
Choose an action ===> 1

Type length of stack (<100): 10

-----------Menu----------
| 1) Make random stack  |
```

```
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 12 57 47 47 7 19 40 58 61 43 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 5

Top value = 12

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 4

12 was popped!

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 57 47 47 7 19 40 58 61 43 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
```

```
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 7
-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 7 19 40 43 47 47 57 58 61 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 3

Type value: 51

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 51 7 19 40 43 47 47 57 58 61 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
```

```
Choose an action ===> 3

Type value: 34

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 34 51 7 19 40 43 47 47 57 58 61 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 7
-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 7 19 34 40 43 47 47 51 57 58 61 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 6

Size of stack = 11

-----------Menu----------
| 1) Make random stack  |
```

```
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 2

[ 7 19 34 40 43 47 47 51 57 58 61 ]

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 4

7 was popped!

-----------Menu----------
| 1) Make random stack  |
|    2) Print stack     |
|     3) Push value     |
|     4) Pop value      |
|   5) Read top value   |
|   6) Size of stack    |
|  7) Sorting of stack  |
|     8) Print Menu     |
|        9) Exit        |
-------------------------
Choose an action ===> 9
```

9. Дневник отладки должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечание автора по существу работы _____

_____

11. Выводы Я научился автоматизировать сборку программ модульной структуры на языке Си с использованием утилиты make.

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом _____

_____

Подпись студента _____