|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра прикладной математики | | | |
|  | | | |
| Практическое задание № 2 | | | |
| по дисциплине «Структуры данных и алгоритмы» | | | |
| **СТРУКТУРЫ ДАННЫХ СТЕКИ и ОЧЕРЕДИ** | | | |
|  | | | |
|  | Группа | ПМ-95 |
| Бригада | 8 |
| Алгоритм | Лаворов Дмитрий |
| Реализация | Костыря Сергей |
| Тестирование | Краус александр |
| Преподаватель | Тракимус Юрий Викторович |
| Дата | 22.05.2020 |
|  | | | |
| Новосибирск | | | |

**1. Условие задачи**

Написать операции работы с заданной структурой данных, включив их в один модуль (файл). К основным операциям добавить операцию, показывающую содержимое структуры после выполнения какого-либо действия с ней. Эту операцию реализовать на основе базовых операций: основные операции над статическим стеком.

**2. Анализ задачи**

Дано: статический стек, заданный в виде структуры.

Результат: демонстрация 5 операций, выполняющих действия над статическим стеком.

**3. Метод решения**

Для демонстрации каждой из 5 операций над статическим стеком используются 6 подпрограммы, каждая соответственно отвечает за осуществление своей операции (проверка на пустоту, извлечение элемента, проверка на заполненность, добавление элемента, просмотр всего стека). Пользователю предлагается ввести одну из команд, осуществив нужную ему операцию, после чего результат выполнения этой операции будет выведен на экран в виде изменённого стека.

**4. Структура входных и выходных данный**

Внешнее представление входных данных: команды в виде чисел от 0 до 5, демонстрирующие выполнение операций над структурой.

Внутреннее представление входных данных: массив целочисленных значений, описанный в структуре, struct stack { int arr[N]; int count; }; изменяющийся после операций.

Внутреннее представление результата:

массив целочисленных значений, описанный в структуре struct stack { int arr[N]; int count; };

Внешнее представление результата: Последний символ в стеке, над которым была совершена операция, весь стек целиком, либо сообщение об ошибке.

**5. Алгоритм решения**

На вход поступает одна из шести команд предложенных пользователю.

Введите «1» для добавления элемента в вершину стека;

Введите «2» для взятия элемента из стека;

Введите «3» для вывода элемента стека;

Введите «4» для проверки пустоты стека;

Введите «5» для проверки заполненности стека;

Введите «0» для выхода;

Добавить элемент: пока значение вершины стека не равно «размерность стека – 10»\* добавляет в стек элемент в ячейку – вершину стека, начиная с ячейки - 0 и заканчивая ячейкой 10. При попытке добавить элемент в заполненный стек выводится сообщение “Стек полный”

Проверить стек: int Full(stack\* ps) – проверка стека на заполнение, если в стеке есть эле-мент(ы), но он не полный – выводится “Есть возможность добавить %d элементов”, если стек пуст – “Стек пуст”, если стек заполнен – “Стек полный”.

Извлечь элемент: void Take(stack\* ps, int\* result) – проверяет пуст ли стек, если нет, то выводит элемент на экран и переносит вершину стека на предыдущий элемент, за счет функции ps->count— тем самым забывая извлеченный элемент, иначе вызывает сообщение «Стек пуст»

Распечатать стек: void Show(stack\* ps) – проверяет пуст ли стек, если нет, то выводит его на экран.

Выход из программы: break – при нажатии клавиши с цифрой 0, то программа досрочно прекращает работу и закрывается.

**6. Текст программы**

#include <stdio.h>

#include <conio.h>

#include <locale.h>

const int N = 10;

struct stack

{

int arr[N];

int count;

};

//Изначальная очистка стека

void Initial(stack\* ps)

{

ps->count = 0;

}

//Проверка стека на пустоту

int Empty(stack\* ps)

{

if (ps->count == 0)

return 1;

else

return 0;

}

//Функция взятия элемента из стека

void Take(stack\* ps, int\* result)

{

\*result = (ps->arr)[(ps->count) - 1];

ps->count--;

}

// Функция проверки заполненности стека

int Full(stack\* ps)

{

if (ps->count == N)

return 1;

else

return 0;

}

// Функция добавление элемента в стек

void Add(stack\* ps)

{

scanf\_s("%d", &(ps->arr[ps->count]));

ps->count++;

}

//Функция просмотра стека

void Show(stack\* ps)

{

for (int i = 0; i < ps->count; i++)

printf(" %d", ps->arr[i]);

}

// Вывод команд для работы со стеком

void main()

{

stack s;

setlocale(LC\_ALL, "Russian");

Initial(&s);

int otv;

int m = 0;

printf\_s("Введите «1» для добавления элемента в вершину стека\n");

printf\_s("Введите «2» для взятия элемента из стека\n");

printf\_s("Введите «3» для вывода элемента стека\n");

printf\_s("Введите «4» для проверки пустоты стека\n");

printf\_s("Введите «5» для проверки заполненности стека\n");

printf\_s("Введите «0» для выхода\n"); printf\_s("0. Выход\n");

printf\_s(" Что требуется выполнить? Введите цифру: ");

do

{

scanf\_s("%d", &otv);

switch (otv)

{

case 1:

if (Full(&s) == 1)

printf\_s("Стек полный\n");

else

{

Add(&s);

printf\_s("Элемент добавлен в стек\n");

}

break;

case 2:

if (!Empty(&s))

{

Take(&s, &m);

printf\_s("Из стека извлечён элемент: %d \n", m);

}

break;

case 3:

if (Empty(&s) == 1)

printf\_s("Стек пуст\n");

else

{

printf\_s("Элементы стека");

Show(&s);

printf\_s("\n");

}

break;

case 4:

if (Empty(&s) == 1)

printf\_s("Стек пуст\n");

else

printf\_s("В стеке %d элемента\n", s.count);

break;

case 5:

if (Full(&s) == 1)

printf\_s("Стек полный\n");

else

printf\_s("Есть возможность добавить %d элементов\n", N - s.count);

break;

case 0:

break;

default:

printf\_s("Ошибка");

break;

}

}

while (otv != 0);

\_getch();

}

**7. Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Дано/Команды** | **Результат** | **Примечание** |
| **1** | 1 36 | Элемент добавлен в стек | Добавление элемента |
| **2** | 1 256 2 | Из стека извлечён элемент 256 | Взять элемент |
| **3** | 1 67 3 | Элементы стека 67 | Вывести элемент стека |
| **4** | 1 1999 4 | В стеке 1 элемент | Проверка пустоты стека |
| **5** | 1 999999 5 | Есть возможность добавить ещё 9 элементов | Проверка заполненности стека |
| **6** | 0 | Программа успешно закрывается | Проверка команды выход |