1. Считать строку символов char. Обойти строку при помощи указателей и преобразовать большие буквы в маленькие, а маленькие в большие. Остальные символы оставить без изменения.

2. Модифицировать программу так, чтобы строки считывались из файла in.txt и преобразованные данные записывались в файл out.txt.

3. Создать массив из 100 элементов. Заполнить его случайными данными в диапазоне от 10 до 99. Для обхода массива использовать указатели. Найти в массиве максимальный элемент и подсчитать количество элементов массива, равных максимальному. Поиск максимального элемента и подсчёт количества реализовать за один проход по массиву.

4. В файле хранится строка, содержащая арифметическое выражение из целых чисел, а также знаков +, -, *, / без скобок. Вычислить значение выражения.

5. Объявить динамический массив. Длину массива вводить с клавиатуры. Заполнить массив случайными числами и найти максимальную по длине последовательность подряд идущих элементов массива. К элементам массива **не** обращаться через квадратные скобки. Использовать указатели.

6. Дан двумерный динамический массив. Создать два новых массива на основании исходного. Первый отсортировать по возрастанию суммы элементов строк. Второй отсортировать по убыванию суммы элементов столбцов. К элементам массива **не** обращаться через квадратные скобки. Использовать указатели.

Стек

Стеком (англ. *stack*) называется хранилище данных, в котором можно работать только с одним элементом: тем, который был добавлен в стек последним. Стек должен поддерживать следующие операции:

push

Добавить (положить) в конец стека новый элемент

pop

Извлечь из стека последний элемент

back

Узнать значение последнего элемента (не удаляя его)

size

Узнать количество элементов в стеке

clear

Очистить стек (удалить из него все элементы)

Хранить элементы стека мы будем в массиве. Для начала будем считать, что максимальное количество элементов в стеке не может превосходить константы MAX_SIZE, тогда для хранения элементов массива необходимо создать массив размера MAX_SIZE.

Объявим структуру данных типа stack.

```
const int MAX_SIZE=1000;

struct stack {
   int m_size;          // Количество элементов в стеке
   int m_elems[MAX_SIZE]; // Массив для хранения элементов

   stack();             // Конструктор
   ~stack();            // Деструктор
   void push(int d);    // Добавить в стек новый элемент
   int pop();           // Удалить из стека последний элемент
                        // и вернуть его значение
   int back();          // Вернуть значение последнего элемента
   int size();          // Вернуть количество элементов в стеке
   void clear();        // Очистить стек
};
```

Объявленная здесь структура данных stack реализует стек целых чисел. Поле структуры m_size хранит количество элементов в стеке в настоящее время, сами элементы хранятся в элементах массива m_elems с индексами 0..m_size-1. Элементы, добавленные позже, получают большие номера.

7. Упражнение A - простой стек

Реализуйте структуру данных "стек", реализовав все указанные здесь методы. Напишите программу (функцию main), содержащую описание стека и моделирующую работу стека. Функция maincчитывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения одной команды программа должна вывести одну строчку. Возможные команды для программы:

push n

Добавить в стек число n (значение n задается после команды). Программа должна вывести ok.

pop

Удалить из стека последний элемент. Программа должна вывести его значение.

back

Программа должна вывести значение последнего элемента, не удаляя его из стека.

size

Программа должна вывести количество элементов в стеке.

clear

Программа должна очистить стек и вывести ok.

exit

Программа должна вывести bye и завершить работу.

Гарантируется, что набор входных команд удовлетворяет следующим требованиям: максимальное количество элементов в стеке в любой момент не превосходит 100, все команды pop_back и back корректны, то есть при их исполнении в стеке содержится хотя бы один элемент.

Пример протокола работы программы

Ввод	Вывод
push 2	ok
push 3	ok
push 5	ok
back	5
size	3
pop	5
size	2
push 7	ok
pop	7
clear	ok
size	0
exit	bye

8. Упражнение B - стек с обработкой ошибок

Аналогично предыдущему заданию, только снимается ограничение на корректность вызовов методов back и pop. Данные операции должны перед исполнением проверять, содержится ли в стеке хотя бы один элемент. Если во входных данных встречается операция back или pop, при этом стек пуст, то программа должна вместо числового значения вывести строку error.

При этом должна быть реализована двойная защита: вызов методов forward и pop для пустого стека не должен приводить к обращению к несуществующим элементам массива m_elems, а функция main должна выводить сообщение error, при считывании некорректной операции.

Пример протокола работы программы

Ввод	Вывод
push 2	ok
back	2
pop	2
size	0
pop	error
push 1	ok
size	1
exit	bye

9. Упражнение C - стек без ограничения на размер

Реализуйте стек динамического размера, то есть ограниченный только объемом свободной оперативной памяти. Для этого используйте указатели и динамически распределяемую память. Если для полностью заполненного стека вызывается метод push размер динамического массива, отведенного для хранения стека, должен увеличиваться.

10. Арифметическое выражение можно представить в обратной польской записи, где знаки операции следуют за операндами, а не ставятся между ними, как в обычной записи выражений. Обратная польская запись не требует скобок. Например, выражению 1+2 соответствует запись 1 2 +, а выражению 1 +2*3 соответствует запись 1 2 3 * +, запись (2+3) * (3-1) записывается как 2 3 + 3 1 - *. Задаётся строка – выражение в обратной полькой записи (числа и знаки +, -, * разделены пробелами). Используя стек, вычислите значение выражения.

11. Дана последовательность скобок вида (,), [,], {, }. Правильными скобочными последовательностями называются пустая последовательность, а также (P), [P], {P}, где P – это некоторая правильная последовательность. Например, {}()[] и {[][()()]}() – правильные скобочные последовательности, а (], [{{}, ({}} – неправильные. Определите является ли заданная строка правильным скобочным выражением.

12. Реализовать класс _String, предоставляющий функционал работы со строками. Интерфейс класса:

```
using namespace std;
class _String {
    char *value;
    int stringLength = -1;
public:
    _String() {
    }
    _String(char *val) {
    }
    _String(const char *val) {
    }
    ~_String() {
    }
    _String(const _String &element) {
    }
    int length() {
    }
    void append(_String str) {
    }
    void append(char *str) {
    }
    void append(const char *str) {
    }
    void setValue(char *val) {
    }
    void setValue(const char *val) {
    }
    void setValue(int val) {
    }
    void setValue(double val, int precision = 10) {
    }
```

```
    _String operator = (const _String &element) {
    }
    _String operator + (const _String &right) {
    }
    int pos(const char *str) {
    }
    void remove(int position, int length) {
    }
    void insert(int position, int length, const char *source) {
    }
    void insert(int position, int length, char *source) {
    }
    void insert(int position, int length, _String source) {
    }
    _String copy(int position, int length) {
    }
    char operator [] (int index) {
    }
    int parseInt(int &error) {
    }
    float parseFloat(int &error) {
    }
    bool isEqual(char *str) {
    }
    bool isEqual(const char *str) {
    }
};
```

13. Классы _String и stack вынести в отдельные файлы.

14. Используя класс _String и stack, реализовать построение постфиксной записи арифметического выражения, содержащего знаки операций +, -, *, / и скобки. Учитывать возможные ошибки в расставленных скобках.

15. Реализовать класс Parses, который выполняет преобразование входного выражения в выражение в постфиксной форме.

16. Реализовать класс, который принимает на вход выражение в постфиксной форме и выполняет расчет. Возвращает результат типа вещественного типа.

17. Реализовать класс, который выполняет полную обработку выражения: вычисляет значение выражения со скобками.

18. Создать систему классов для реализации pattern command для вычисления значения арифметических выражений. Система классов должна поддерживать операции сложения, вычитания, умножения, деления, вычисление значений тригонометрических функций, функции максимума, минимума.

19. Реализовать pattern factory method для порождения классов из задания 18.