Лабораторная №7. Рекурсивные алгоритмы для массивов. Обработка ошибок.

Темы, рассматриваемые в лабораторной работе:

Рекурсивная реализация функций обработки массивов. Использование исключений в функциях, работающих с массивами. Использование двух указателей для работы с сегментом массива.

Пример рекурсивной реализации функции поиска максимума в одномерном массиве:

```
int arr_max(const int* a, int length) {
    if (length <= 1)
        return a[0];
    return max(a[0], arr_max(a + 1, length - 1));
}
. . .
cout << "MAX: " << arr_max(aaa, n)<< endl;
```

Пример функции, использующей исключение для сигнализации об ошибке:

```
void arr_change(int* a, int length, int number) {
  if ((number<0)||(length<1)|| (number>=length)) throw  1000;
      a[number] = -1;
}
// int poisk_zero(const int* a, int length);
//поиск первого нуля в массиве
. . .

try{
   arr_change(aaa , n , poisk_zero(aaa,n));
}catch(int e)
{
  cout<< " в массиве нет нулей, массив не изменился "
      << endl;
}
```

Пример функции, работающей с диапазоном элементов в массиве:

```
int count_zero(const int * beg, const int * end){
int k=0;
      for (const int* p = beg; p < end; ++p)
            if  (*p==0) k++;
return k;
}
. . .
// примеры вызова
//  count_zero(aaa, aaa+n) – весь массив из n элементов
//  count_zero(aaa, aaa+2) – первые 2
//  count_zero(aaa+n-4, aaa+n) – последние 4
```

Задания:

Решение каждой задачи должно сопровождаться представительным набором тестов.

1. Реализовать рекурсивное вычисление суммы элементов одномерного массива.
2. Реализовать рекурсивное определение количества элементов с заданным свойством (свойство задается параметром – функцией).
3. Реализовать рекурсивное определение значения максимального элемента в сегменте массива
4. Реализовать рекурсивно проверку свойства - массив не содержит нулей. Функция должна возвращать логическое значение.
5. Функция (не рекурсивная) должна определять отношение среднего арифметического элементов массива до первого нуля к среднему арифметическому элементов после первого нуля. Предусмотреть проверку всех ситуаций, когда вычисление невозможно.
6. Дополнить функцию ввода динамического массива

```
void read(int *&a, size_t &n)
```

проверкой, что динамический массив еще не размещен в памяти.