МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«Витебский государственный технологический университет»

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Методические указания по выполнению расчетно-графических работ
для студентов специальности
1-40 05 01 «Информационные системы и технологии»,
направления специальности
1-40 05 01-01 «Информационные системы и технологии
(в проектировании и производстве)»

Витебск
2018

УДК 681.5 (075.8)

Составитель:

В. Е. Казаков

Рекомендовано к изданию редакционно-издательским советом
УО «ВГТУ» протокол № 7 от 28.09.2018.

Основы алгоритмизации и программирования: методические указания по выполнению расчетно-графических работ / сост. В. Е. Казаков. – Витебск : УО «ВГТУ», 2018. – 28 с.

Методические указания являются руководством по выполнению расчетно-графических работ по курсу «Основы алгоритмизации и программирования», определяют общие требования, предъявляемые к их выполнению, оформлению и содержанию, освещают последовательность ее подготовки и процедуру защиты.

УДК 681.5 (075.8)

СОДЕРЖАНИЕ

Расчётно-графическая работа 1. Алгоритмы обработки основных структур данных

Общие требования к выполнению, отчёту и защите

Разработанный студентом программный продукт должен решать поставленную задачу и обладать всеми свойствами алгоритма (конечность, результативность и т. д.), не приводить к возникновению ошибок во время исполнения при вводе различных наборов входных данных.

Блок-схема разработанного алгоритма должна соответствовать исходному тесту программы.

Программа должна реализовывать пользовательский интерфейс, обеспечивающий необходимое количество запусков вычислительной части с различными входными данными, также при наличии нескольких функциональных возможностей необходимо предоставить пользователю возможность их выбора при работе программы.

При разработке программы необходимо придерживаться принципа модульности: реализовывать слабосвязанные, повторяющиеся алгоритмы в отдельных функциях, размещать логически связанные группы функций в отдельных модулях.

Для хранения больших объёмов данных в программе необходимо использовать динамические структуры.

Отчёт о решении каждой задачи должен содержать:

1) полный текст задания;
2) блок-схему алгоритма, указанного в задании, выполненную в соответствии с требованиями ГОСТ 19.701–90 (ИСО 5807–85);
3) исходный текст программы;
4) результаты тестовых запусков.

На защите студент должен по заданию преподавателя:

– описать назначение каждой из переменных;

– описать назначение каждой части алгоритма;

– объяснить, как будет изменяться работа программы при вводе различных данных;

– обосновать выбор метода решения задачи;

– указать возможные пути и методы улучшения показателей качества программы (скорость выполнения, требуемый объём оперативной памяти, понятность и удобство для модификации исходного кода программы);

– предложить альтернативный метод решения задачи.

Методика выполнения заданий

Задание 1

Задан одномерный массив. Вставить перед отрицательными элементами элементы с нулевым значением. Добавление элемента в массив реализовать при помощи функции. В первую очередь, разработаем функцию для вывода массива в консоль. В функцию передаётся указатель на вещественное значение (массив вещественных значений) и размер массива.

```
void print_arr (float *X, int n)
{
  printf("\n");
  for (int i = 0;i<n;i++) printf("%.1f   ",X[i]);
}
```

Разработаем функцию вставки в массив на заданную позицию нового элемента с нулевым значением. В качестве дополнительной возможности предусмотрим изменение переменной, содержащей размер массива, а также вставку в заданную позицию любого значения.

Внутри функции мы создадим новый массив, то есть передаваемый указатель будет изменяться, поэтому параметр функции: указатель на указатель. Также принимаем адрес переменной, в которой хранится размер массива (значение этой переменной мы изменим внутри функции), позицию, в которую будет вставлено значение, и само значение.

```
int insert_el (float ** X, int &n, int insInd, int insEl)
{
        float * N = new float [n+1];      // выделяем память под новый массив
        memcpy(N,*X,n*sizeof(float)); // копируем в него исходный массив
        delete []* X;                          // удаляем исходный массив
// сдвигаем элементы после позиции вставки
        for (int i = n-1;i>=insInd;i--)
         N[i+1] = N[i];
// записываем вставляемое значение в позицию вставки
        N[insInd] = insEl;
        n++;            // увеличиваем переменную с размером массива
       *X = N; // записываем ссылку на новый массив в переменную исходного массива
        return 0;
}
```

Разработанные функции поместим в модуль Arr (Arr.h, Arr.cpp) и разместим его в корне папки проекта.

Разработаем программу, решающую задачу с использованием разработанных функций.

```
#include "Arr.h"
#define N 5

int _tmain(int argc, _TCHAR* argv[])
{
// для отладки можно использовать массив, инициализированный в коде программы
float s[N] = {-1,2,-3,4,-5};
        float *b = new float [N];
        memcpy(b,s,sizeof(float)*N);
        int   n = N;
// выводим исходный массив
        print_arr (b, n);
        for (int i = 0; i < n; i++) { // перебираем массив
          if (b[i]<0)                 // если находим отрицательный элемент
                {
                    insert_el (&b, n, i, 0);  // вставляем 0 перед ним
                    i++;                      // переходим к следующему элементу
                }
        }
        print_arr (b, n);
        return 0;
}
```

Протестируем работу приложения. Тестовый пример позволяет проверить работу программы при наличии отрицательного элемента в начале, в конце и в середине массива.

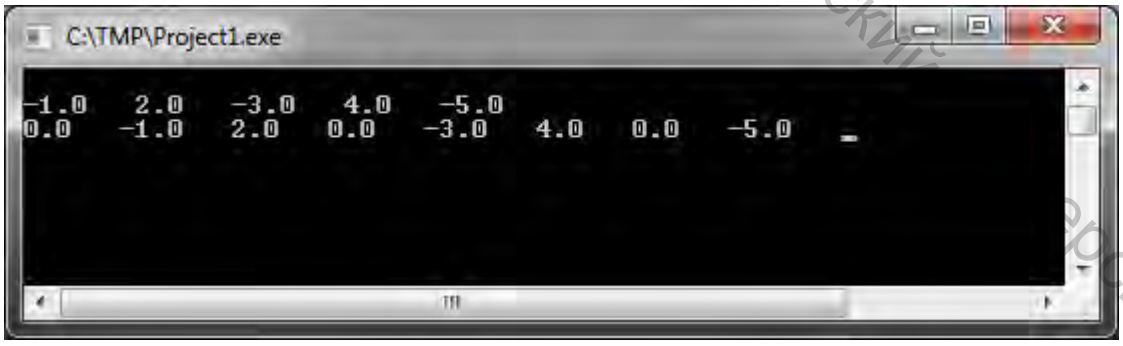

Рисунок 1 – Результат запуска программы по заданию 1

Какие ещё тесты нужно провести для данной программы?

Задание 2

Дано целое число. Подсчитать сумму цифр, составляющих данное число. Привести блок-схему алгоритма реализации функции вычисления суммы цифр числа.

В качестве дополнительной возможности реализуем нахождение суммы цифр и положительного и отрицательного числа.

```
int zsum(int x)
{
    int s = 0;
    if ( x<0 ) x = -x;     // если число отрицательное, то меняем его знак
    while (x)
    {
        s += x % 10;     // остаток от деления на 10 выделяет младший разряд числа
        x /= 10;         // целочисленное деление на 10 удаляет младший разряд числа
    }
    return s;
}
```

Разработаем блок-схему алгоритма функции.

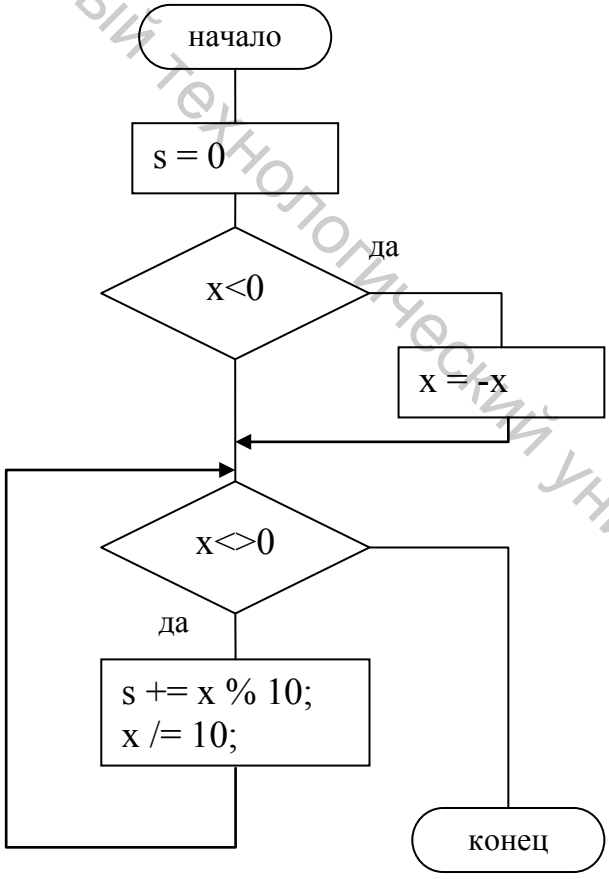

Рисунок 2 – Блок-схема алгоритма функции

Разработаем основную программу со следующим интерфейсом: пользователь вводит число – программа возвращает сумму цифр и выводит подсказку «<ESC> - Exit», пользователь может нажать <ESC> и выйти из программы, либо нажать на любую другую клавишу и продолжить работу.

```
int _tmain(int argc, _TCHAR* argv[])
{
    int ch;

    do {
     printf(">");
     scanf("%i",&ch);                  // вводим число
     printf(" = %i\n",zsum(ch));       // выводим сумму цифр
     printf("<ESC> - Exit\n");         // выводим подсказку
    }  while (getch() != 27);          // если нажат <ESC>

    return 0;
}
```

Протестируем работу приложения.

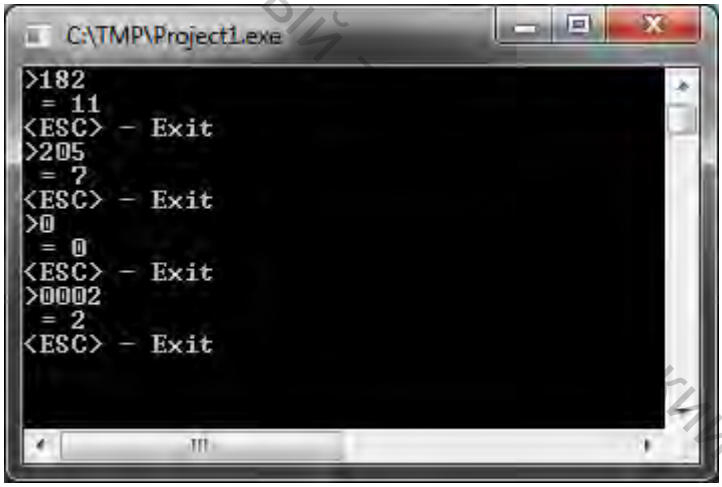

Рисунок 3 – Результат запуска метода отображения дерева

Задание 3

Имеется файл, содержащий информацию о точках на плоскости (x, y). Разработать функции: создания, вывода на экран и редактирования таких файлов.

Вначале разработаем структуру для хранения координат точки.

```
# define FILE_BEG 0
# define MAX_FILE_NAME 100
```

```
struct Point {
  float x,y;
 };
```

Разработаем функцию вывода имени файла.

```
void getFileName(char * fn){
  printf("Enter file name> ");
  gets(fn);
 }
```

После каждой операции с файлом он будет закрываться.

Разработаем функцию вывода содержимого файла на экран. Функция принимает название файла.

```
int prnFile(char *fn)
 {
   FILE *F;
   Point P1;
   if (!(F = fopen(fn,"r")))        // пытаемся открыть файл с полученным именем
// невозможно открыть файл
          {   printf("Can`t open the file\n"); return 1 ; }
   int i = 1;
          while(fread(&P1,sizeof(Point),1,F))  // читаем файл
                printf("%i: %f \t %f \n",i++,P1.x,P1.y);
          fclose(F);
          return 0;
 }
```

Разработаем функцию создания и заполнения файла. Функция принимает название файла.

```
 int crFile (char *fn)
 {
  FILE *F;
  Point P1;
   if (!(F = fopen(fn,"w")))
          {      printf("Can`t create the file\n"); return 1 ;}
          do {
                printf("x y >");
                scanf("%f %f",&P1.x,&P1.y);   //считываем координаты точки
                 fwrite(&P1,sizeof(Point),1,F);  // записываем их в файл
                printf("press \"Esc\" - for exit \n");
          } while(getch() != 27);   // завершаем, когда пользователь нажмёт <Esc>
```

```
        fclose(F);
}
```

Разработаем функцию редактирования файла. Функция принимает название файла.

```
int editFile(char *fn)
{
 FILE *F;
 Point P1;
 if (!(F = fopen(fn,"r+")))
     {       printf("Can`t open the file\n"); return 1 ;}
 printf("Enter record number \n");
 int id;
 scanf("%i",&id);   // запрашиваем номер редактируемой записи
 printf("x y >");
 scanf("%f %f",&P1.x,&P1.y);
 fseek(F, sizeof(Point)*(id-1), 0);  // указатель перед ней
 fwrite(&P1,sizeof(Point),1,F); // перезаписываем запись
 fclose(F);
}
```

Основная программа будет реализовывать простой интерфейс: выбор действия по нажатию кнопки: 1 – создание файла, 2 – вывод файла, 3 – редактирование файла, 0 – выход из программы.

```
int _tmain(int argc, _TCHAR* argv[])
{
  char fn[MAX_FILE_NAME];
        do {
              printf("1-create, 2-print, 3-edit, 0-escape\n");

              switch (getch()) {
              case '1':
                    getFileName(fn);
                    crFile (fn);
                    break;
              case '2':
                    getFileName(fn);
                    prnFile (fn); break;
              case '3':
                   getFileName(fn);
                   editFile (fn); break;
              case '0':
                   return 0;
```

```
                default:
                        printf("Unknown command\n");
                }
        } while(1);
  return 0;
}
```

Протестируем работу приложения.

Рисунок 4 – Результат запуска программы по заданию 3

Задания к расчётно-графической работе 1

Задание 1

Общие требования:

– использовать динамические массивы;

– в основной программе разработать пользовательский интерфейс, обеспечивающий доступ к разработанной функции.

Вариант 1. Дана целочисленная матрица размера M на N. Найти номер последнего из ее столбцов, содержащих равное количество положительных и отрицательных элементов (нулевые элементы матрицы не учитываются). Ес-

ли таких столбцов нет, то вывести 0. Проверку равенства количества положительных и отрицательных элементов реализовать с помощью функции.

Вариант 2. Задан символ C и строка S. Удвоить каждое вхождение символа C в строку S. Разработать функцию для поиска символа в строке, возвращающую номер позиции следующего вхождения символа C в строку S.

Вариант 3. Описать функцию, которая формирует по данной строке S массив слов, входящих в S. Словом считается набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки. Используя эту функцию, найти количество слов N, содержащихся в данной строке S, и сами эти слова.

Вариант 4. Описать функцию, формирующую для вещественного массива A размера N массив индексов элементов, не превосходящих среднеарифметическое всех элементов массива.

Вариант 5. Задан одномерный массив A. Удалить из массива повторяющиеся элементы. Удаление элемента из массива реализовать при помощи функции.

Вариант 6. Дана целочисленная матрица размера M на N. Реализовать функцию, меняющую местами верхнюю и нижнюю половины матрицы.

Вариант 7. Задан одномерный массив A. Получить массив B, заполнив его индексами чётных элементов массива A. Поиск очередного чётного элемента выполнять при помощи функции.

Вариант 8. В двумерном массиве A размером 2 на N заданы координаты точек, составляющих некоторый выпуклый многоугольник. Описать функцию, находящую длину отрезка AB на плоскости по координатам его концов: $|AB| = ((xA - xB)^2 + (yA - yB)^2)^{1/2}$ (xA, yA, xB, yB – вещественные параметры). С помощью этой функции найти периметр заданного многоугольника.

Вариант 9. В двумерном массиве вычислить сумму элементов в каждом столбце и каждой строке. Вычисление суммы в строке, столбце выполнить при помощи одной функции.

Вариант 10. Описать процедуру, выполняющую транспонирование (то есть зеркальное отражение относительно главной диагонали) квадратной вещественной матрицы A произвольного размера.

Задание 2

Общие требования:

– разработать функцию согласно заданию;

– расширить возможности функции (например, проверка правильности входных параметров функции);

– в отчёте привести блок-схему алгоритма реализации функции;

– в основной программе разработать пользовательский интерфейс, обеспечивающий доступ к разработанной функции.

Вариант 1. Дана строка. Если она представляет собой запись целого числа, то вывести 1, если вещественного (с дробной частью) – вывести 2; если строку нельзя преобразовать в число, то вывести 0. Считать, что дробная часть вещественного числа отделяется от его целой части десятичной точкой «.» или запятой «,».

Вариант 2. Дана строка, содержащая, по крайней мере, один символ пробела. Вывести подстроку, расположенную между первым и вторым пробелом исходной строки. Если строка содержит только один пробел, то вывести пустую строку.

Вариант 3. Дана строка, представляющая целое положительное число. Получить новое число путём удаления из исходного одной цифры из заданной позиции (например: 12345–1245).

Вариант 4. Дано целое положительное число. Вывести символы, изображающие цифры этого числа в порядке справа налево.

Вариант 5. Дана строка, содержащая полное имя файла. Выделить из этой строки путь.

Вариант 6. Дана строка, состоящая из слов (словом считать группу символов, начинающуюся и заканчивающуюся пробелом). Вывести слова, содержащие численные значения.

Вариант 7. Дана строка, состоящая из слов (словом считать группу символов, начинающуюся и заканчивающуюся пробелом, или знаками препинания). Вывести самое длинное слово.

Вариант 8. Дана строка, состоящая из слов (словом считать группу символов, начинающуюся и заканчивающуюся одним или несколькими пробелами). Вывести строку, содержащую эти же слова, разделенные одним символом «.» (точка). В конце строки точку не ставить.

Вариант 9. Дана строка, состоящая из слов (словом считать группу символов, начинающуюся и заканчивающуюся одним или несколькими пробелами). Вывести количество символов в каждом слове.

Вариант 10. Дана строка, содержащая полное имя файла. Выделить из этой строки название последнего каталога (без символов «\»). Если файл содержится в корневом каталоге, то вывести символ «\».

Задание 3

Общие требования:

– разработать функцию создания файла с данными согласно заданию;

– разработать функцию чтения и вывода на экран файла с данными;

– количество элементов входных и выходных файлов заранее не известно;

– в основной программе разработать пользовательский интерфейс, обеспечивающий доступ к разработанному функционалу.

Вариант 1. Имеется файл, содержащий целые числа. Создать два новых файла, первый из которых содержит положительные числа из исходного файла (в обратном порядке), а второй – отрицательные (также в обратном порядке). Если положительные или отрицательные числа в исходном файле отсутствуют, то соответствующий результирующий файл оставить пустым.

Вариант 2. Дана действительная квадратная матрица порядка A (n×n). Сохранить в файл ненулевые элементы матрицы, предусмотреть просмотр таких файлов с полным восстановлением исходной матрицы.

Вариант 3. Имеется файл, содержащий целые числа. Продублировать в нем все нечётные элементы.

Вариант 4. Получить значения функции Y(x) на заданном пользователем отрезке с указанным пользователем шагом.

$$Y = \begin{cases} \ln(1+x^2) - 2x, & x < -1 \\ \cos(x+1) - 2cd, & -1 \le x < 1 \\ x^3 - cd, & x \ge 1 \end{cases}$$

Предусмотреть возможность сохранения полученной таблицы в файл и просмотра файлов с такими таблицами (при просмотре необходимо показывать значения констант, отрезок табулирования и шаг).

Вариант 5. Сохранить в файл три одномерных массива разной размерности, заданных пользователем. Предусмотреть возможность просмотра содержимого таких файлов с полным восстановлением каждого из массивов.

Вариант 6. Имеется файл, содержащий вещественные числа. Заменить в файле каждый элемент, кроме начального и конечного, на среднее арифметическое трёх элементов: его самого, предыдущего и последующего элемента.

Вариант 7. Имеется файл, содержащий целые числа. Удалить из него все элементы, расположенные на позициях с четными номерами.

Вариант 8. Имеется файл, содержащий целые числа. Удвоить его размер, записав в конец файла все его исходные элементы (в обратном порядке).

Вариант 9. Разработать программу сохранения в файл двумерного массива произвольного размера. Предусмотреть возможность просмотра содержимого таких файлов с полным восстановлением содержащихся в них массивов.

Вариант 10. Имеется файл, содержащий целые числа. Заменить в нем каждый отрицательный элемент на два нуля.

Расчётно-графическая работа 2. ООП и динамические структуры

Общие требования к разработке, отчёту и защите

Разработанный студентом программный продукт должен решать поставленную задачу, с использованием объектно-ориентированного подхода, не приводить к возникновению ошибок во время исполнения при вводе различных наборов входных данных.

Программа должна реализовывать пользовательский интерфейс, обеспечивающий необходимое количество запусков вычислительной части с различными входными данными, также при наличии нескольких функциональных возможностей необходимо предоставить пользователю возможность их выбора при работе программы.

При разработке программы необходимо придерживаться принципа модульности: размещать разработанные классы в отдельных модулях.

Для хранения элементов структуры необходимо использовать динамическое распределение памяти.

Необходимо предусмотреть корректное завершение работы программы (освобождение использованной динамической памяти).

Дополнительные задания:

1) предложить оригинальный подход к решению задачи;
2) предусмотреть обработку некорректных значений входных данных.

Отчёт о решении задачи должен содержать:

1) полный текст задания;
2) описание разработанной структуры данных;
3) описание методов разработанных классов;

4) результаты тестовых запусков;
5) исходный текст программы.

На защите студент должен по заданию преподавателя:
– обосновать выбор метода решения задачи;
– описать назначение и алгоритм реализации каждого из методов разработанного класса;
– объяснить, как будет изменяться работа программы при вводе различных входных данных;
– предложить альтернативный метод решения задачи.

Методика выполнения заданий

Рассмотрим следующее задание.

Разработать класс для работы с динамической структурой «Бинарное поисковое дерево», содержащей целочисленные значения.

Реализовать:
1) параметрический конструктор,
2) деструктор,
3) метод добавления элемента в дерево,
4) метод удаления элемента и его ветви из дерева,
5) просмотр содержимого дерева.

Решить задачу с использованием разработанного класса: задано множество элементов дерева. Подсчитать количество элементов с чётными ключами.

Структура хранения данных бинарного дерева будет представлять собой набор узлов, каждый из которых имеет две ссылки на дочерние узлы. Для хранения всей структуры дерева необходимо создать класс, который будет содержать ссылку на корневой элемент.

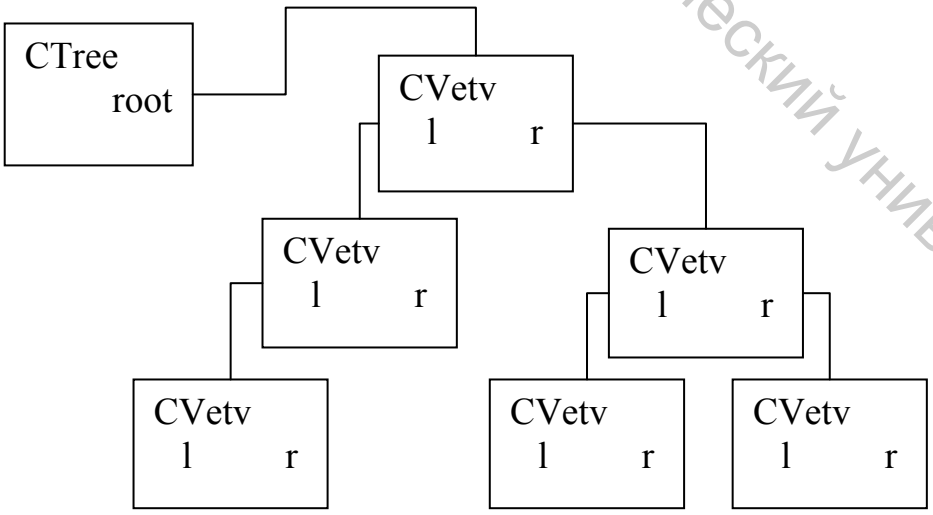

Рисунок 5 – Структура хранения данных

Разработаем **класс узла структуры**:

```
class CVetv
{
        public:
        int key;                    // поле ключа
         int inf;                   // поле информационного значения
        CVetv *l,*r;                // указатели не левую и правую ветвь
        CVetv (int n, int key)      // параметрический конструктор
        {
                this->inf = inf;
                this->key = key;
                l = r = NULL;
        }
};
```

Разработаем **класс структуры**.

```
class CTree
{
       private:
        CVetv *root;          // указатель на корневой узел
        public:
          CTree(){            // конструктор
                root = NULL;
        }
};
```

Разработка методов класса. Для каждого метода необходимо определить входные параметры и возвращаемое значение. Кроме того, необходимо определить последовательность создания методов – вначале создаются базовые методы (методы, которые могут быть использованы для реализации других методов).

Метод добавления элемента

По параметру метод будет принимать ключ узла и его значение, возвращать метод будет указатель на созданный узел. Поскольку это поисковое дерево, то добавляться элемент будет в определённую ветвь. Кроме того, ключ добавляемого узла должен быть уникальным.

```
CVetv* CTree::add(int n_inf, int n_key)
  {
        int     find = 0;     // признак нахождения дубликата нового ключа
        CVetv *       prev;
        CVetv * t = root;
```

```
        // пока не найден дубликат, или не вышли за пределы дерева
    while ( t && ! find) {
        prev = t ;
        if( n_key == t->key)
        find = 1;                    // найден дубликат добавляемого ключа
        else
                                     // переходим по левой или правой ветке
          if ( n_key < t -> key ) t = t -> l;
              else   t = t -> r;
    }
    if (!find) {                                  // найдено место с адресом prev
        t = new CVetv(n_inf,n_key);   // создаем новый узел
        if ( n_key < prev -> key )    // и присоединяем его, либо
                    prev -> l = t;    // к левой,
        else    prev -> r = t;        // или к правой ветви
        return t;
    }
    else
     return NULL;            // есть дубликат
}
```

Параметрический конструктор нашего класса будет загружать пары «ключ-значение» из файла следующего формата:

```
10 20
12 32
6 44
```

В качестве параметра метод будет принимать название файла.

```
CTree::CTree( char* fn)
{
    FILE * f;
    if(!(f = fopen(fn,"rt")))          // проверка наличия файла
    {
        printf("file not found");
        return;
    }
    char ts[255] = "";
    fgets(ts,255,f);
    int inkey,ininf;
                                  // считываем и создаём корневой узел
    sscanf_s(ts,"%i %i",&inkey, &ininf);
    CVetv * t = new CVetv(ininf,inkey);
    root = t;
                                  // считываем строки из файла и добавляем в
                                  // дерево узлы при помощи метода add
```

```
        while(fgets(ts,255,f)) {
          sscanf_s(ts,"%i %i",&inkey, &ininf);
          add(ininf,inkey);
          }
          fclose(f);
}
```

Метод удаления ветви. Метод принимает указатель на ветвь, которую необходимо удалить. Реализация будет выполнена с использованием рекурсии.

```
void CTree::del_vet(CVetv *d)
{
 if ( d ) {              // если полученный указатель не «пустой»
   del_vet( d ->r); d ->r = NULL; // вызываем этот же метод для правого узла
   del_vet( d ->l); d ->l = NULL;  // и для левого узла
        delete d;                  // удаляем узел
 }
}
```

Деструктор. Для реализации деструктора достаточно воспользоваться уже разработанным методом **del_vet**.

```
CTree::~CTree ()
{
 del_vet( root->r);  // вызываем метод del_vet для правой ветви корневого узла
 del_vet( root->l);  // вызываем метод del_vet для правой ветви корневого узла
        delete root;  // удаляем корень
 }
```

Метод отображения дерева. В метод передаётся указатель на узел-владелец отображаемой ветки и уровень узла, который будет использоваться для формирования отступа, с которым прорисовывается узел. Реализация метода также будет рекурсивной.

```
void CTree::view ( CVetv *t, int lv )
{
 if ( t ) {                      // если полученный указатель не «пустой»
        view ( t ->r , lv+1);    // вывод правой ветви
        // формирование отступа от начала строки в
        // соответствии с уровнем
        for ( int i=0; i<lv; i++)  printf("    ");
        printf(" %i\n", t -> key);
        view( t ->l , lv+1);     // вывод левой ветви
 }
}
```

Добавим ещё один метод для **отображения всего дерева**. Метод не будет иметь параметров.

```
void CTree::view_all ()
{
        view(root,0);
}
```

Рисунок 6 – Результат запуска метода отображения дерева

Метод вычисления количества узлов с чётными индексами

В метод передаётся указатель на узел-владелец ветки, в которой необходимо подсчитать количество элементов с чётными индексами. Метод возвращает найденное значение количества элементов с чётными индексами.

Реализация метода схожа с реализацией метода отображения дерева – в обоих случаях необходим перебор всех узлов дерева с помощью рекурсии.

```
int CTree::odd_cnt (CVetv* t)
{
 int c = 0;
 if ( t ) {                    // если полученный указатель не «пустой»
  // суммируем количество требуемых значений в правой ветке
  c += odd_cnt(t ->r);
  // проверяем собственный индекс
  if ( fmod(t->key,2) ==0 ) c++;
 // суммируем количество требуемых значений в левой ветке
  c += odd_cnt (t ->l);
 }
 return c;
}
```

Разработанные классы необходимо разместить в отдельном модуле (CTree.h, CTree.cpp) и подключать этот модуль по необходимости к основной программе, или к другому модулю.

Предложите другой подход к разработке методов, обоснуйте своё решение.

Разрабатываем консольное приложение, используя разработанный модуль.

```
#include <conio.h>
#include "CTree.h"

int _tmain(int argc, _TCHAR* argv[])
{
   CTree* t = new CTree ("Data.txt");
   t->view_all();
   printf("number of odd elements = %i",t->odd_cnt());
   delete t;
   getch();
   return 0;
}
```

Для тестирования проекта создадим несколько тестовых примеров.

Создадим файл с данными для инициализации дерева Data.txt и проведём несколько тестовых запусков программы с различными входными данными.

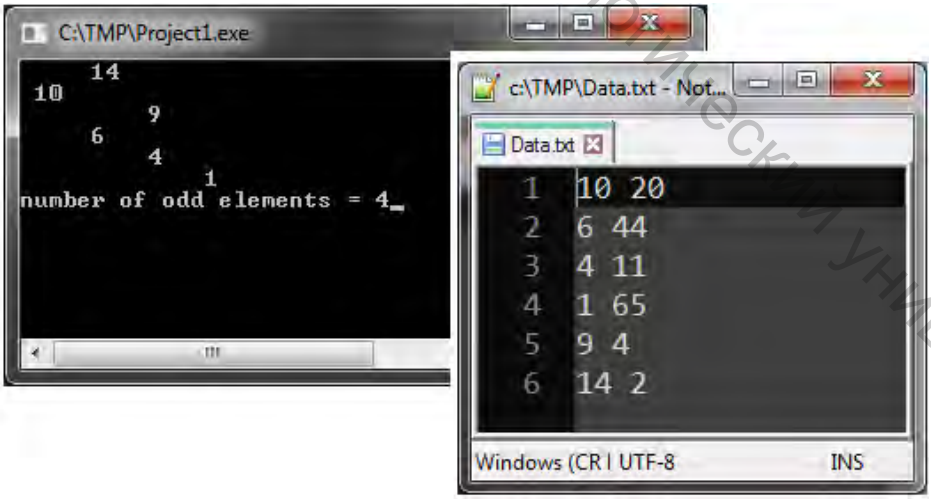

Рисунок 7 – Результат тестирования приложения 1

Как видим данный тестовый пример отработан программой правильно.

Разработаем другой пример: проверим, как работает программа в случае наличия узлов с дублирующимися ключами и в случае отсутствия ключей с чётными номерами.

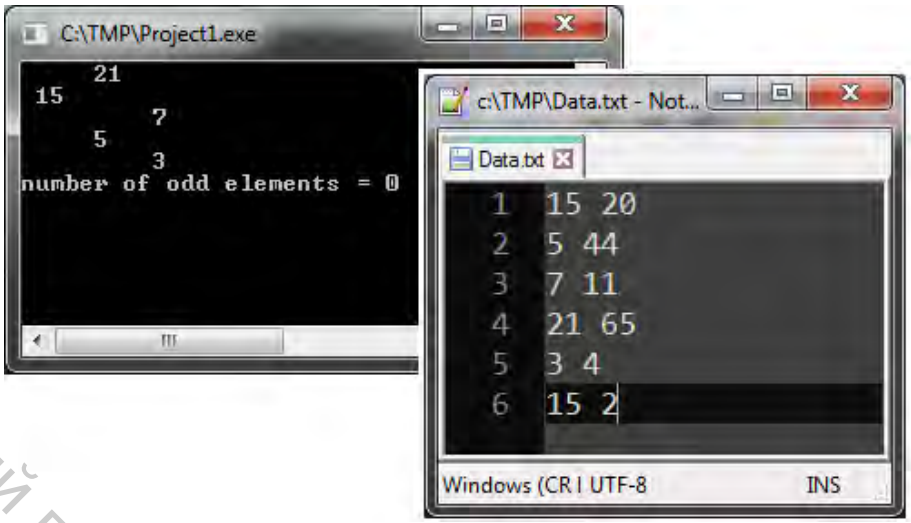

Рисунок 8 – Результат тестирования приложения 2

Как видим, узел «15-2» не попал в дерево, и программа данный тестовый пример отработала правильно.

Предложите ещё несколько вариантов тестирования.

Задания к расчётно-графической работе 2

ВАРИАНТ 1

Задание 1. Разработать класс для работы с динамической структурой «Очередь», содержащей целочисленные значения.

Реализовать:

1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод постановки элемента в очередь,
5) метод выборки элемента из очереди,
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: задана последовательность целых чисел. Получить все чётные элементы последовательности.

ВАРИАНТ 2

Задание 1. Разработать класс для работы с динамической структурой «Стек», содержащей целочисленные значения.

Реализовать:

1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,

4) метод добавления элемента в стек,
5) метод извлечения элемента из стека,
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: задана строка с математическим выражением, определить, правильно ли в ней расставлены скобки. Пример: (1+2)*(1+(2-2)*4) – правильно; (1-2)*(2*(2+1) – не правильно.

ВАРИАНТ 3

Задание 1. Разработать класс для работы с динамической структурой «Однонаправленный список», содержащей целочисленные значения.

Реализовать:
1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод добавления элемента в хвост списка,
5) метод удаления элемента из хвоста списка,
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: задана последовательность целых чисел. Записать в структуру исходные значения в обратном порядке.

ВАРИАНТ 4

Задание 1. Разработать класс для работы с динамической структурой «Закольцованный список», содержащей целочисленные значения.

Реализовать:
1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод добавления элемента в любое место списка,
5) метод удаления указанного элемента списка,
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: С помощью последовательности уникальных целых чисел (номера остановок) задан закольцованный маршрут движения транспорта (например: 3, 5, 1, 6, 8, 9, 2, 3), определить, в каком направлении нужно двигаться, чтобы быстрее добраться от остановки с номером а до остановки с номером б.

ВАРИАНТ 5

Задание 1. Разработать класс для работы с динамической структурой «Очередь», содержащей вещественные значения.

Реализовать:

1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод постановки в очередь элемента,
5) метод выборки элемента из очереди.
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: заданы две упорядоченные последовательности. Объединить их в одну общую упорядоченную последовательность.

ВАРИАНТ 6

Задание 1. Разработать класс для работы с динамической структурой «Стек», содержащей вещественные значения.

Реализовать:

1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод добавления элемента в стек,
5) метод извлечения элемента из стека.
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: задана последовательность целых чисел. Вычислить среднеарифметическое отрицательных значений.

ВАРИАНТ 7

Задание 1. Разработать класс для работы с динамической структурой «Однонаправленный список», содержащей вещественные значения.

Реализовать:

1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод добавления элемента в хвост списка,
5) метод удаления элемента из хвоста списка,
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: задана последовательность целых чисел. Записать в структуру исходные значения в обратном порядке.

ВАРИАНТ 8

Задание 1. Разработать класс для работы с динамической структурой «Закольцованный список», содержащей вещественные значения.

Реализовать:

1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод добавления элемента в любое место списка,
5) метод удаления указанного элемента списка,
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: реализовать алгоритм «считалочки»: пользователь последовательно вводит целые числа (n), необходимо удалять из структуры элемент, который находится через n элементов от текущего положения.

ВАРИАНТ 9

Задание 1. Разработать класс для работы с динамической структурой «Очередь», содержащей целочисленные значения.

Реализовать:

1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод постановки в очередь элемента,
5) метод выборки из очереди элемента,
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: задана последовательность целых чисел. Удалить из данной последовательности все нечётные числа.

ВАРИАНТ 10

Задание 1. Разработать класс для работы с динамической структурой «Стек», содержащей целочисленные значения.

Реализовать:

1) конструктор по умолчанию,
2) параметрический конструктор,
3) деструктор,
4) метод добавления элемента в стек,

5) метод извлечения элемента из стека,
6) просмотр содержимого структуры.

Задание 2. Решить задачу с использованием разработанного класса: заполнить исходный стек случайными значениями, разделить значения между двумя новыми стеками (в первый поместить все чётные значения, во второй – нечётные).

Список рекомендуемой литературы

1. Дейтел, Х. М. Как программировать на С / Х. М. Дейтел, П. Дж. Дейтел. – М.: Бином, 2006. – 1037 с.
2. Страуструп, Б. Язык программирования С++ / Б. Страуструп. – М., СПб.: Издательство БИНО – Невский диалект, 2001. – 1099 с.
3. Йодан, Э. Структурное программирование и конструирование программ / Э. Йодан. – М.: Мир, 1979.
4. Кормен, Т. Алгоритмы: Построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – СПб, 2003.
5. Бусько, В. Л. Основы алгоритмизации и программирования в среде Visual С++ : лаб. практикум по курсу «Основы алгоритмизации и программирования» для студ. 1–2-го курсов всех спец. БГУИР / В. Л. Бусько, А. А. Навроцкий. – Минск : БГУИР, 2008. – 66 с.

Учебное издание

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Методические указания по выполнению расчетно-графических работ

Составитель:
Казаков Вадим Евгеньевич

Редактор *Н.В. Медведева*
Корректор *Т.А. Осипова*
Компьютерная верстка *В.Е. Казаков*

Подписано к печати 11.10.2018. Формат 60x90 $^{1}/_{16}$. Усл. печ. листов 1,8.
Уч.-изд. листов 2,0. Тираж 50 экз. Заказ № 289.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 72.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1497 от 30 мая 2017 г.