МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Случайные БДП – вставка и исключение. Текущий контроль

Студент гр. 7383 _____ Корякин М.П.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Корякин М.П.

Группа 7383

Тема работы: Случайные БДП – вставка и исключение. Текущий контроль.

Исходные данные:

Написать программу, генерирующую задания по случайному БДП.

Содержание пояснительной записки:

- Содержание
- Введение
- Формулировка задачи
- Решение задачи
- Тестирование
- Заключение
- Приложения А. Исходный код программы

Предполагаемый объем пояснительной записки не менее 15 страниц.

Дата выдачи задания: 19.10.2018

Дата сдачи курсовой работы:

Дата защиты курсовой работы:

Студент гр.7383 _____ Корякин М.П.

Преподаватель _____ Размочаева Н.В.

АННОТАЦИЯ

В курсовой работе была реализована программа, генерирующая варианты заданий по теме «Случайные БДП», а также прорешивающая все эти задания.

SUMMARY

In the course work was implemented a program that generates variants of tasks on the topic "Random BDP", as well as solve all these tasks.

Содержание

ВВЕДЕНИЕ

Целью работы является практическое освоение стандартных алгоритмов статического кодирования и декодирования.

Задачей является создать программу, генерирующую задания по теме «Случайные БДП» - вставка и исключение.

Тестирование будет проводится в OS Linux Mint 18.01.

1. ФОРМУЛИРОВКА ЗАДАЧИ

Написать программу для генерирования вариантов заданий и их решения по теме «Случайные БДП» - вставка и исключение, а также сортировка элементов БДП по возрастанию.

1.2. Теоретические сведения

- Структура случайного БДП полностью зависит от того порядка, в котором элементы расположены во входной последовательности (во входном файле).
- Исключение элемента из случайного БДП. Исключить элемент из случайного БДП проще всего, если этот элемент находится в листе дерева. Тогда данный лист непосредственно удаляется. Если же удаляемый элемент находится во внутреннем узле дерева, то следует найти минимальный элемент правого поддерева этого узла, если правое поддерево существует, как указано на рис. 1.

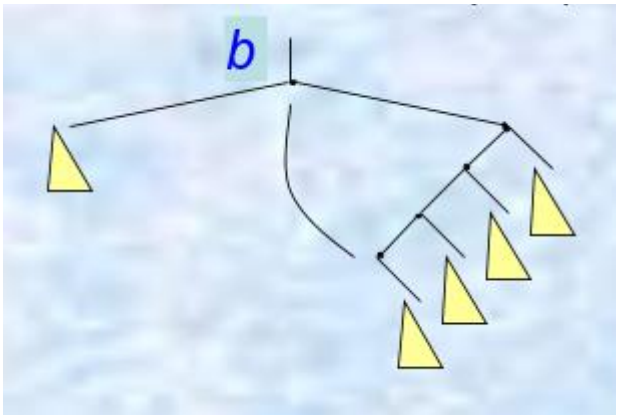


Рисунок 1 – удаление элемента, который находится во внутреннем узле дерева, когда у удаляемого элемента существует правое поддерево.

В случае, когда правого поддерева удаляемого элемента не существует, то следует найти максимальный элемент левого поддерева, как показано на рис. 2.

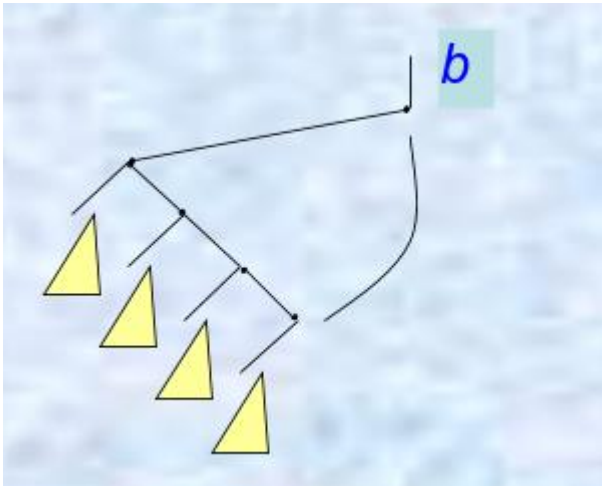


Рисунок 2 – удаление элемента, который находится во внутреннем узле дерева, когда у удаляемого элемента не существует правого поддерева.

- Вставка элемента в случайное БДП. Вставка элемента построенное случаное БДП происходит путем следованию по самому дереву, сравнивая листы дерева. Тем самым новый вставленный элемент становится листом прежнего БДП.

2. РЕШЕНИЕ ЗАДАЧИ

В классе BST реализованы такие приватные поля, как:

- Key – значение ключа элемента.
- Right – указатель элемента на правое поддерево.
- Left – указатель элемента на левое поддерево.

А также в данном классе реализованы публичные методы, такие как:

- BST – инициализация элемента.
- Int Root – возвращает значение ключа элемента.
- BST* destroy – удаляет случайное БДП.
- BST* Left – возвращает указатель на левое поддерево элемента.
- BST* Right – возращает указатель на правое дерево элемента.
- BST* BuildBST – осуществляет строительство дерева, по заданному ему элементу.
- BST* find – осуществляет поиск элемента в дереве с определенным ключом.
- BST* remove – осуществляет исключение элемента из дерева с определенным ключом.
- BST* join – осуществляет склейку левого и правого поддерева, исключенного элемента.

Помимо созданных методов были использованы такие функции:

- Void printtree – осуществляет вывод иллюстрации построенного дерева в файл с ответами.
- Int comp – компоратор для сортировки массива ключей элементов по возрастанию, используется для решения задачи с построением дерева в порядке возрастания.
- Int main – главная функция, которая инициализирует в консоли небольшой диалог с пользователем, спрашивая у него количество требуемых вариантов заданий, типы заданий, которые будут

укомплектованы в варинтах заданий. После чего, используя вышеописанные методы и функции, генерирует нужное количество вариантов с выбранными пользователем типом заданий.

Код программы представлен в приложении А.

3. ТЕСТИРОВАНИЕ

Тестирование программы проводилось в OS Linux Mint 18.01. Использованные во время работы программы файлы были размещены в той же папке, что и запускаемый файл программы.

3.1. Первый запуск программы.

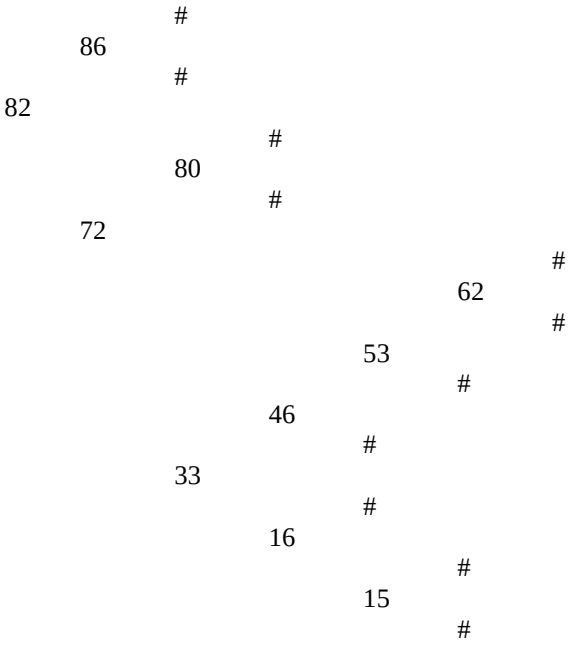
```
Введите кол-во вариантов заданий
3
Введите кол-во узлов дерева
10
Дерево #1 сгенерировано
Если вы хотите отрисовать дерево по возрастанию - введите 1
1
Если вы хотите исключить какой-либо элекмент - введите 2
2
Если вы хотите вставить какой-либо элемент - введите 3
3
Дерево #2 сгенерировано
Дерево #3 сгенерировано
```

Рисунок 3 – консольное меню при первом тестовом запуске программы.

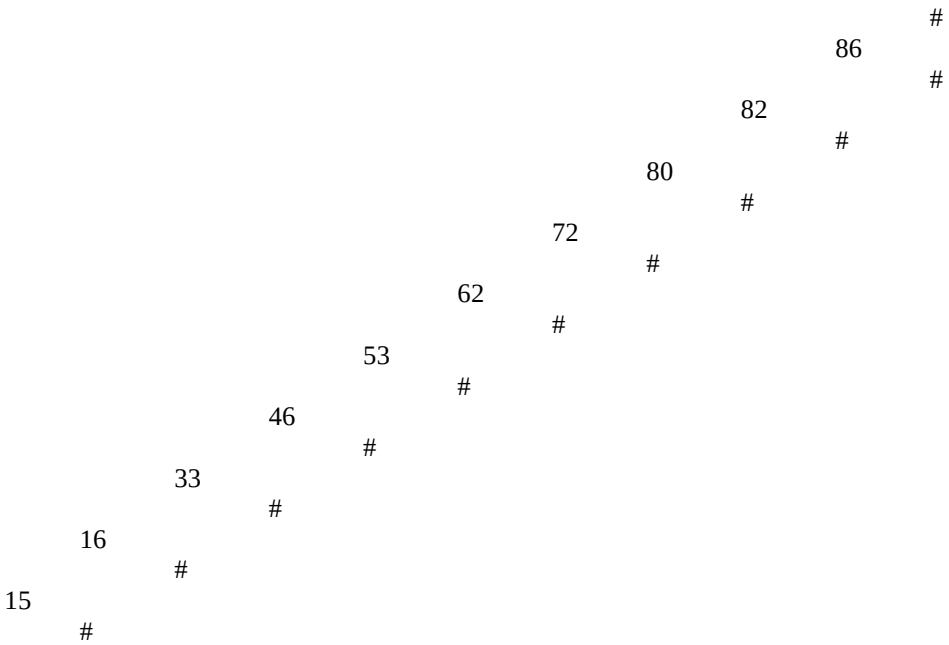
```
Вариант №1
Ваше случайное дерево:       82 72 33 46 16 80 86     53 62 15
Отрисуйте дерево по возрастанию
Исключить из дерева 53 элемент и нарисовать дерево
Вставить в дерево 90 элемент и нарисовать дерево
Вариант №2
Ваше случайное дерево:       19 9 46 29 73 89 37 11 32 98
Отрисуйте дерево по возрастанию
Исключить из дерева 29 элемент и нарисовать дерево
Вставить в дерево 62 элемент и нарисовать дерево
Вариант №3
Ваше случайное дерево:       79 18 24 5 75    1 53 30 81 56
Отрисуйте дерево по возрастанию
Исключить из дерева 30 элемент и нарисовать дерево
Вставить в дерево 99 элемент и нарисовать дерево
```

Рисунок 4 – содержимое файла task.txt после первого запуска.

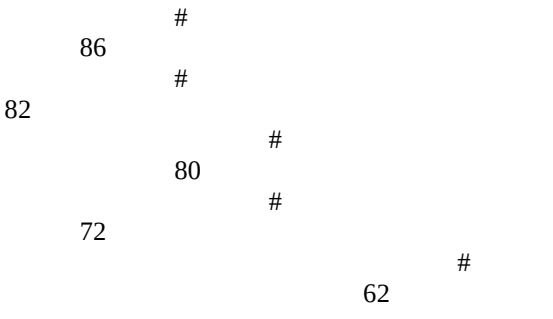
Вариант №1
Иллюстрация сгенерированного дерева



Дерево перестроено по возрастанию:



Дерево перестроено с исключением элемента:



#
46
#
33
#
16
#
15
#

Дерево перестроено с добавлением элемента:

#
90
#
86
#
82
#
80
#
72
#
62
#
46
#
33
#
16
#
15
#

Вариант №2
Иллюстрация сгенерированного дерева

#
98
#
89
#
73
#
46
#
37
#
32
#
29
#
19
#
11
#
9
#

Дерево перестроено по возрастанию:

#

98
#
89
#
73
#
46
#
37
#
32
#
29
#
19
#
11
#
9
#

Дерево перестроено с исключением элемента:

#
98
#
89
#
73
#
46
#
37
#
32
#
19
#
11
#
9
#

Дерево перестроено с добавлением элемента:

#
98
#
89
#
73
#
62
#
46
#
37
#
32
#
19
#

```
            11
                #
        9
            #
```

Вариант №3
Иллюстрация сгенерированного дерева

```
            #
        81
            #
79
                    #
                75
                        #
                    56
                        #
                53
                        #
                    30
                        #
            24
                #
        18
                #
            5
                    #
                1
                    #
```

Дерево перестроено по возрастанию:

```
                                                #
                                            81
                                                #
                                        79
                                            #
                                    75
                                        #
                                56
                                    #
                            53
                                #
                        30
                            #
                    24
                        #
                18
                        #
            5
                #
1
        #
```

Дерево перестроено с исключением элемента:

```
            #
        81
            #
79
                    #
                75
```

```
                                        #
                                    56
                                        #
                                53
                                    #
            24
                    #
    18
                    #
            5
                                #
                    1
                                #
```

Дерево перестроено с добавлением элемента:

```
                    #
            99
                    #
    81
            #
79
                                #
                    75
                                        #
                                    56
                                        #
                                53
                                    #
            24
                    #
    18
                    #
            5
                                #
                    1
                                #
```

Рисунок 5 – содержимое файла answ.txt после первого запуска.

3.2. Второй запуск программы.

```
Введите кол-во вариантов заданий
1
Введите кол-во узлов дерева
0
Дерево #1 сгенерировано
Если вы хотите отрисовать дерево по возрастанию - введите 1
1
Если вы хотите исключить какой-либо элекмент - введите 2
2
Если вы хотите вставить какой-либо элемент - введите 3
3
```

Рисунок 6 – консольное меню после второго запуска.

```
Вариант №1
Ваше случайное дерево:
Отрисуйте дерево по возрастанию
Вставить в дерево 35 элемент и нарисовать дерево
```

Рисунок 7 – содержимое файла task.txt после второго запуска.

```
_____
Вариант №1
Иллюстрация сгенерированного дерева

#

Дерево перестроено по возрастанию:

#

Дерево перестроено с добавлением элемента:

        #
35
        #
```

Рисунок 8 – содержимое файла answ.txt после второго запуска.

3.3. Третий запуск программы.

```
Введите кол-во вариантов заданий
0
```

Рисунок 9 – консольное меню после третьего запуска.

После третьего запуска программы в файлы task.txt и answ.txt ничего не было записано.

ЗАКЛЮЧЕНИЕ

В ходе работы была реализована программа на языке C++, которая взаимодействует с пользователем с помощью консольного меню, генерируя заданное количество вариантов на тему «Случаное БДП», а также с заданным количеством узлов дерева.

Текущий контроль осуществлен с помощью записи вариантов заданий в файл, а ответов в другой файл.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <vector>
#include <cstring>
#include <fstream>
#include <sstream>
#include <cstdlib>
using namespace std;
class BST{
private:
    int key;
    BST* right;
    BST* left;
public:
    BST(int k){
        key = k;
        left = right = nullptr;}
    int Root(BST* b){
        if (b == NULL)
            exit(1);
        else
            return b->key;
    }
    BST* destroy(BST* tree){
            if (left)
                 delete tree->left;
            if (right)
                    delete tree->right;
            delete tree;
            return tree = NULL;}
    BST* Left(BST* b){
        if (b == NULL)
        exit(1);
        else
        return b->left;}
    BST* Right(BST* b){
        if (b == NULL)
        exit(1);
        else
        return b->right;}
    BST* BuildBST(BST* tree, int keyint){
        if (!tree)
            return new BST(keyint);
        if( tree->key > keyint )
                    tree->left = BuildBST(Left(tree), keyint);
            else
                    tree->right = BuildBST(Right(tree), keyint);
        return tree;}
```

```
BST* find( BST* tree, int key){
    if(!tree)
        return NULL;
    if(key == tree->key)
        return tree;
    if(key < tree->key)
        return find(tree->left, key);
    else
        return find(tree->right, key);}
BST* remove(BST* p, int k, int gl){
     int ppp;
     if (k==gl){
                BST* j=p;
                if(j->right){
                     j=j->right;
                     while(j->left)
                          j=j->left;
                }
                ppp = j->key;
     }
if( !p ) return p;
     if( p->key==k){
        BST* q = join(p->left,p->right);
        p=q;
           if (k==gl && p){
                p->key=ppp;
           }
        return p;
    }
    else if( k<p->key )
        p->left = remove(p->left,k, gl);
    else
        p->right = remove(p->right,k, gl);
    return p;
}
BST* join(BST* L, BST* R){
     if( !R ) return L;
     if( !L ) return R;
     BST* h = R;
     BST* k;
     if (!h->left){
          h->left=L;
          return h;
     }

     while(h->left){
          k=h;
          h=h->left;
     }
     if (h->right){
          k->left=h->right;
```

```
          }
          else k->left=NULL;
          h->right=R;
          h->left=L;
          return h;

     }
};
void printtree(BST* treenode, int l, ofstream &file){
   if(treenode==NULL){
       for(int i = 0;i<l;++i)
           file<<"\t";
       file<<'#'<<endl;
       return;
   }
   printtree(treenode->Right(treenode), l+1, file);
   for(int i = 0; i < l; i++)
       file << "\t";
   file << treenode->Root(treenode)<< endl;
   printtree(treenode->Left(treenode),l+1, file);}
int comp (const int *i, const int *j)
{
return *i - *j;
}
int main(){
    BST *b = NULL;
    BST *p = NULL;
    string exp;
    stringbuf temp;
    int i;
    int call, j;
    j=0;
    cout<<"Введите кол-во вариантов заданий"<<endl;
    cin>>call;
    cin.ignore();
    int c1=0, c2=0, c3=0;
    int vozr, foriskl, forvstav, iskl, vstav;
    ofstream file;
    file.open("1.txt");
    ofstream answ;
    answ.open("answ.txt");
    int arr[100];
    int ch=0;
while(call>j){
    j++;
    if (!ch){
          cout<<"\nВведите кол-во узлов дерева"<<endl;
    cin>>ch;
    cin.ignore();
    srand(ch);
    }
```

```
          srand( time(0)+j );
          cout<<"Дерево #"<<j<<" сгенерировано"<<endl;
          for(i=0; i<ch; i++){
              arr[i]=1+rand() % 99;
          }
          file<<"\nВариант №"<<j;
          file<<"\nВаше случайное дерево:\t";
          for(i=0; i<ch; i++){
              file<<arr[i]<<"\t";
          }
          answ<<"_____\n";
          answ<<"Вариант №"<<j;
          answ<<"\nИллюстрация сгенерированного дерева\n\n";
          for(i=0; i<ch; i++){
               if(b->find(b, arr[i])){
                           continue;
              }
              b = b->BuildBST(b, arr[i]);
          }
          printtree(b, 0, answ);
          if(c1==0){
          cout<<"\nЕсли вы хотите отрисовать дерево по возрастанию -
введите 1"<<endl;
               cin>>vozr;
          cin.ignore();
               c1=1;
          }
          if(vozr==1){
               file<<"\nОтрисуйте дерево по возрастанию";
               qsort(arr, ch, sizeof (int), (int(*) (const void *,
const void *)) comp);
               for (i=0; i<ch; i++){
                         if(p->find(p, arr[i]))
                         continue;
                         p = p->BuildBST(p, arr[i]);
              }
               answ<<"\nДерево перестроено по возрастанию:\n\n";
               printtree(p, 0, answ);
          }
          if (c2==0){
          cout<<"\nЕсли вы хотите исключить какой-либо элекмент -
введите 2"<<endl;
               //int iskl;
          cin>>iskl;
          cin.ignore();
               c2=1;
          }
          if(iskl==2){
               foriskl=arr[0+rand() % ch];
               file<<"\nИсключить из дерева "<<foriskl<<" элемент и
```

```
нарисовать дерево";
                b=b->remove(b, foriskl, arr[0]);
                answ<<"\nДерево перестроено с исключением
элемента:\n\n";
                printtree(b, 0, answ);
          }
          if (c3==0){
                cout<<"\nЕсли вы хотите вставить какой-либо элемент -
введите 3"<<endl;
                cin>>vstav;
                cin.ignore();
                c3=1;
          }
          if(vstav==3){
                srand(time(NULL)+arr[j]);
                forvstav=1+rand() % 99;
                file<<"\nВставить в дерево "<<forvstav<<" элемент и
нарисовать дерево";
                BST* k = k->BuildBST(b, forvstav);
                answ<<"\nДерево перестроено с добавлением
элемента:\n\n";
                printtree(k, 0, answ);
          }
          b = b->destroy(b);
          p = p->destroy(p);
      }
      file.close();
      answ.close();
      return 0;
      }
```