МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ | **Факультет прикладной математики и информатики**

Кафедра прикладной математики

Практическое задание № 3

по дисциплине «Структуры данных и алгоритмы»

БИНАРНЫЕ ДЕРЕВЬЯ

Бригада 2	ТАДЖИБАЕВ ЗАВКИДДИН
Группа ПМ-25	ЯГОДКИН АЛЕКСАНДР
Вариант 1 л	СУХАРЕВА СОФЬЯ

Преподаватель ТРАКИМУС ЮРИЙ ВИКТОРОВИЧ

Новосибирск, 2024

1 Задание

Используя очередь или стек (считать уже описанными их типы и операции над ними), опишите на С тип бинарное дерево (считая, что все элементы имеют некоторый простой тип) и реализуйте в виде процедуры или функции одну из перечисленных ниже операций над бинарным деревом.

Используйте наиболее подходящий для решения задачи обход дерева (в глубину: префиксный / инфиксный / постфиксный, в ширину):

Л) напечатать элементы дерева по уровням: сначала из корня, затем из вершин, дочерних по отношению к корню, затем из вершин, дочерних по отношению к этим вершинам, и т. д. при этом отделить элементы одного уровня от элементов другого.

2 Анализ задания

Входные данные: В файл in.txt вводятся различные целочисленные значения, которые являются элементами дерева, где первое число - это корень.

Выходные данные: В файл out.txt выводятся результаты программы: элементы дерева по уровням: сначала из корня, затем из вершин, дочерних по отношению к корню, затем из вершин, дочерних по отношению к этим вершинам, и т. д. при этом отделить элементы одного уровня от элементов другого. “Не удалось открыть файл out.txt.”; “Не удалось открыть файл in.txt.”; “Дерево пусто."

Решение задачи: После ввода данных в in.txt программа создаёт дерево, потом в функции output мы вызываем BFS (обход в ширину) для записи дерева по уровням.

3 Программа

```
#include <stdio.h>
#include <windows.h>
#include <locale.h>

struct tree
{
   char elem;
   tree *left, *right;
   tree(char _elem = 0, tree *_left = NULL, tree *_right = NULL) :
      left(_left), right(_right), elem(_elem) { };
};

tree *BuildTree(FILE *f)
{
   char c = 0;
   fscanf_s(f, "%c", &c, 1);
   switch (c)
   {
      case '(': {
                  char c = 0;
                  fscanf_s(f, "%c", &c, 1);
                  return new tree(c, BuildTree(f), BuildTree(f));
                }
      case ',': return BuildTree(f);
      case '0': return NULL;
```

```
      default:;
   }
   return NULL;
}


bool input(tree *&t)
{
   FILE *f = NULL;
   fopen_s(&f, "in.txt", "r");
   if (f)
   {
      t = BuildTree(f);
      fclose(f);
      return true;
   }
   else
      return false;
}

struct queue
{
   tree *data;
   queue *next, *beg, *end;

   queue(tree *_data = NULL, queue *_next = NULL) :
      next(_next), data(_data), beg(NULL), end(NULL) { };

   void push(tree *c)
   {
      queue *p = new queue(c);
      end = (empty() ? beg : end->next) = p;
   }

   tree *pop()
   {
      if (empty())
         return NULL;
      tree *r = beg->data;
      queue *p = beg;
      beg = beg->next;
      delete p;
      return r;
   }

   UINT size()
   {
      int k = 0;
      for (queue *p = beg; p; p = p->next, k++);
      return k;
   }
```

```
   inline bool empty() { return beg == NULL; }

} *q = NULL;

void BFS(tree *t, FILE *f)
{
   q = new queue;
   q->push(t);
   for ( ; q->beg; )
   {
      for (int i = q->size(); i; i--)
      {
         tree *h = q->pop();
         fprintf_s(f, "%c ", h->elem);
         if (h->left)
            q->push(h->left);
         if (h->right)
            q->push(h->right);
      }
      fprintf_s(f, "\n");
   }
   delete q;
}

void output(tree *t)
{
   FILE *f = NULL;
   fopen_s(&f, "out.txt", "w");
   if (f)
   {
      BFS(t, f);
      fclose(f);
   }
   else
      printf_s("Не удалось открыть файл out.txt.");
}

int main()
{
   setlocale(0, "");
   UINT cp = GetConsoleCP(), outcp = GetConsoleOutputCP();
   SetConsoleCP(1251);
   SetConsoleOutputCP(1251);
   tree *t = NULL;

   if (input(t))
   {
      if (t)
         output(t);
      else
```

```
        printf_s("Дерево пусто.");
    }
    else
        printf_s("Не удалось открыть файл in.txt.");

    SetConsoleCP(cp);
    SetConsoleOutputCP(outcp);
}
```

4 Набор тестов

№	Входные данные	Назначение
1		Файл in.txt отсутствует.
2		Файл out.txt отсутствует.
3		Файл in.txt пуст.
4	(9,(8,(7,0,0),0),(6,(5,0,(4,0,0)),(3,(2,0,0),(1,0,0))))	Простое бинарные дерево.
5	(A,0,(B,0,(C,0,(D,0,0))))	Бинарное дерево, которое состоит только из правого ветвления
6	(A,(B,(C,(D,0,0),0),0),0)	Бинарное дерево, состоящее только из левого ветвления

5 Результаты работы программы

№	Ввод/Вывод программы
1	Не удалось открыть файл in.txt.
2	Не удалось открыть файл out.txt.
3	Дерево пусто.
4	9 8 6 7 5 3 4 2 1
5	A B C D
6	A B C D