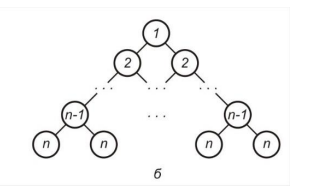
|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра прикладной математики | | | |
| Практическое задание № 3+ | | | |
| по дисциплине «Структуры данных и алгоритмы» | | | |
| **«Древовидные структуры данных»** | | | |
|  | | | |
|  | Группа | ПМ-21 | |
| Студент | Егупов иван | |
|  |  | |
|  | |
|  | |
|  | |
| Преподаватель | ХИЦЕНКО ВАЛЕНТИНА ПАВЛОВНА | |
| Дата | 19.04. | |
| Новосибирск,2023 | | | |

**Условие задачи**

Для бинарного дерева, представленного связным иерархическим списком , используя очередь описать процедуру или функцию, которая по заданному n строит дерево, показанное на рис. 3.5, б.



**Анализ задачи**

Входные данные:

n ∈ N

Выходные данные: S = Дерево на рисунке 3.5, б

Решение:

**при** s2 = 2, i = 1, Q – пустая очередь, p = первый элемент дерева, p = i

**повторять**

***при*** used = s2, положить в очередь первый элемент

***повторять*** p = взять из очереди,

создать левый лист p = i+1, положить его в очередь

создать правый лист p = i+1, положить его в очередь

used = used – 2

***пока*** used≠0

s2 = s2\*2

i = i+1

**пока** i≠n

**Структура программы**

tree\* buildTree(int n) Назначение: строит бинарное дерево по заданному n

int n – заданное значение нижних узлов дерева

Выходные данные: указатель на корень дерева

void treePrint(FILE \*write,tree\* t) Назначение: поуровневая запись элементов дерева в файл

FILE \*write – указатель на файл, tree \*t – указатель на корень дерева

Текст программы

Файл заголовка “Header.h”

#pragma once

#include <stdio.h>

const int N = 20;

struct tree

{

int elem = 0;

tree\* left = nullptr;

tree\* right = nullptr;

};

struct Queue

{

int beg = 0;

int end = -1;

tree\* data[N];

};

int IsEmpty(Queue\* q)

{

int k;

if (q->end == -1) k = 1;

else k = 0;

return k;

}

int Push(Queue\* q, tree\* x)

{

int k = 0;

if (IsEmpty(q))

{

q->beg = 0;

q->end = 0;

q->data[q->beg] = x;

k = 1;

}

else

{

if (q->beg != N - 1)

{

q->beg++;

q->data[q->beg] = x;

k = 1;

}

else

k = 0;

}

return k;

}

tree\* Pop(Queue\* q)

{

tree\* x = nullptr;

if (!IsEmpty(q))

{

x = q->data[q->end];

for (int i = 0; i <= q->beg - 1; ++i)

{

q->data[i] = q->data[i + 1];

}

q->data[q->beg] = NULL;

q->beg--;

if (q->beg < 0)

{

q->beg = 0;

q->end = -1;

}

}

return x;

}

Основная программа Source.cpp

#include <stdio.h>

#include "Header.h"

tree\* buildTree(int n)

{

int s2 = 2;

int i = 1;

Queue q;

q.beg = 0;

q.end = -1;

tree\* d = new tree;

d->elem = i;

Push(&q, d);

while (i!= n)

{

int used = s2;

do

{

tree\* p = Pop(&q);

p->left = new tree;

p->left->elem = i+1;

Push(&q, p->left);

p->right = new tree;

p->right->elem = i+1;

Push(&q, p->right);

used -= 2;

} while (used != 0);

s2 \*= 2;

i++;

}

return d;

}

void treePrint(FILE \*write,tree\* t)

{

Queue q;

q.beg = 0;

q.end = -1;

int k = 1;

int m = 0;

if (t)

{

Push(&q, t);

do

{

tree\* x = Pop(&q);

if (x != nullptr)

{

t = x;

fprintf\_s(write,"%d",t->elem);

k--;

if (t->left)

{

Push(&q, t->left);

m++;

}

if (t->right)

{

Push(&q, t->right);

m++;

}

if (!k)

{

fprintf\_s(write,"\n");

k = m;

m = 0;

}

}

} while (!IsEmpty(&q));

}

}

int main()

{

int n;

scanf\_s("%d", &n);

tree\* root = buildTree(n);

FILE\* write;

fopen\_s(&write, "output.txt", "w");

treePrint(write, root);

return 1;

}

Набор тестов

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Назначение |
| n = 1 | 1 | Простейший тест |
| n = 3 | 1  22  3333 | Проверка работоспособности программы |
| n = 6 | 1  22  3333  44444444  5555555555555555  66666666666666666666 | Еще один тест на проверка работоспособности |