|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TRƯỜNG ĐẠI HỌC GIAO THÔNG VÂN TẢI  KHOA CÔNG NGHỆ THÔNG TIN  ---------------o0o---------------        **Bài tập lớn môn học**      **CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**   |  | | --- | |  |     Giảng viên hướng dẫn: TS.Hoàng Văn Thông  Nhóm thực hiện: Nhóm… Lớp CNTTVA1  Danh sách sinh viên tham gia:   |  |  |  | | --- | --- | --- | | STT | Mã sinh viên | Họ và tên | | 1 | 202602939 | Đinh Thị Hường |   Hà Nội, tháng 12 năm 2020 |

**Nội dung**

**I. Đề bài**

**II. Phân tích bài toán**

1.Yêu cầu của bài toán

2.Các lớp, các thuộc tính, các phương thức của lớp và chức năng của chúng:

2.1.Class Task

2.2.Class Date

2.3.Class BST

**III. Cài đặt các lớp và hàm main bằng C++**

3.1.Code Class Task

3.2.Code Class Date

3.3.Class BST

3.4.File Demo

**IV. Phân tích thời gian chạy của từng phương thức có trong các lớp**

**V. Tài liệu tham khảo**

**I. Đề bài**

**1. Xây dựng lớp cây tìm kiếm nhị phân.**

**2. Viết chương trình quản lý lịch công tác trong tháng đơn giản: cho phép nhập vào nội dung công việc cần làm theo ngày, theo giờ. Trong một ngày có thể có nhiều công việc, mỗi công việc có giờ bắt đầu, tên công việc, nội dung công việc, tính chất công việc {rất quan trọng, quan trọng, bình thường, không cần thiết}, …**

Chương trình có các chức năng chính như sau:Nhập nội dung công việc cần làm theo ngày, theo giờ.

Xem lịch công tác theo ngày yêu cầu.

Xem các công việc theo tính chất: rất quan trọng, quan trọng, …

Xem các công việc đã hoàn tất.

Xem các công việc chưa thực hiện.

Xem các công việc từ ngày a đến ngày b.

Xóa hay điều chỉnh lịch công tác. Nếu sau khi điều chỉnh, ngày nào không còn việc phải làm sẽ xóa khỏi lịch công tác.

**Yêu cầu:** Cấu trúc dữ liệu được sử dụng là cây nhị phân tìm kiếm (BST), trong đó:Mỗi nút trên cây BST là một ngày của lịch công tác.

Trong mỗi nút ngày trên cây lại chứa một danh sách liên kết lưu thông tin các

công việc.

Khi thêm một công việc vào một ngày đã tồn tại trên cây thì công việc này sẽ

được đưa vào danh sách liên kết chứa các công việc theo thứ tự tăng dần của giờ bắt đầu..

**II.Phân tích bài toán**

**1.Yêu cầu của bài toán:**

Sử dụng cấu trúc dữ liệu cây tìm kiếm nhị phân để viết chương trình quản lý công việc trong tháng

**2.Các lớp, các thuộc tính, các phương thức của lớp và chức năng của chúng:**

**2.1. Class Task:**

- Thuộc tính:

+content(string) : Nội dung công việc

+startTime(string): Thời gian bắt đầu công việc

+name(string) : Tên công viêc

+type(string) : Loại công việc(Quan trọng, Rất quan trọng…)

+status(string) : Tình trạng công việc(Đã thực hiên, Chưa thực hiện)

- Phương thức:

+ printTask(): In task theo form.

+ getContent(): Lấy nôi dung công việc.

+ getType(): Lấy loại công việc.

+ getStatus(): Lấy tình trạng công việc.

+ getName(): Lấy tên công việc.

+ getStartTime(): Lấy thời gian bắt đầu công việc.

**2.2. Class Date:**

- Struct: TaskManagement

- Thuộc tính:

+day(int): Ngày trong tháng

+numTask(int): Tổng số công việc

+TaskManagement(TaskManagement): List các công việc

- Phương thức:

+ insert(): Thêm nhiệm vụ vào list các công việc trong ngày.

+ output(): In ra các nhiệm vụ trong ngày.

+ getAmountOfTask(): Lấy ra tổng số công việc trong ngày

+ getDay(): Lấy ra ngày.

+ compareDate(): So sánh ngày khác

+ hasType(): Kiểm tra xem trong danh sách công việc có type này không. Nếu có thì in ra

+ hasStatus(): Kiểm tra xem trong danh sách công việc có status này không. Nếu có thì in ra

+ findTaskByName(): Tìm công việc theo tên.

+ replaceTaskByName(): Điều chỉnh công việc theo tên.

+ removeElementByName(): Xoá công việc theo tên.

**2.3. Class BST:**

- Struct: node

- Thuộc tính:

+root(Node\*): Gốc của cây BST, là nơi quản lí các ngày trong tháng

- Phương thức:

+ insert(): Chèn 1 ngày vào root

+ remove(): Xoá 1 ngày khỏi root.

+ display(): In ra các công việc theo ngày.

+ search(): Tìm kiếm công việc theo ngày.

+ searchByType(): Tìm kiếm công việc theo loại công việc.

+ searchByStatus(): Tìm kiếm công việc tình trạng công việc.

+ searchByDateRange(): Tìm kiếm công việc theo khoảng ngày cho trước

+ replaceTaskByName(): Điều chỉnh công việc theo tên

**III.Cài đặt các lớp và hàm main bằng C++**

**3.1. Code class Task:**

#ifndef TASK\_CPP

#define TASK\_CPP

#include<bits/stdc++.h>

using namespace std;

class Task {

string content, startTime, name, type, status;

public:

Task() {

}

~Task() {

}

Task(string \_startTime, string \_name , string \_content, string \_type, string \_status = "Chua thuc hien") {

this->content = \_content;

this->startTime = \_startTime;

this->name = \_name;

this->type = \_type;

this->status = \_status;

}

void printTask() {

cout << this->startTime << " : " << this->name << " : " << this->content

<< " : " << this->type <<"(" << this->status << ")" << endl;

}

string getContent() {

return this->content;

}

string getType() {

return this->type;

}

string getStatus() {

return this->status;

}

string getName() {

return this->name;

}

string getStartTime() {

return this->startTime;

}

};

#endif

**3.2. Code class Date:**

#ifndef DATE\_CPP

#define DATE\_CPP

#include<bits/stdc++.h>

using namespace std;

class Date {

int day;

int numTask;

struct TaskManagement {

Task\* task;

TaskManagement\* next;

};

TaskManagement\* tm;

public:

Date() {

this->day = -1;

this->numTask = 0;

this->tm = NULL;

}

~Date() {

day = -1;;

this->numTask = 0;

tm = NULL;

}

Date(int \_d) {

this->day = \_d;

this->numTask = 0;

this->tm = NULL;

}

void insert(string startTime, string name, string content, string type) {

TaskManagement\* newTm = new TaskManagement();

newTm->task = new Task(startTime, name, content, type);

newTm->next = NULL;

if (this->tm == NULL) {

this->tm = newTm;

} else if (this->tm->next == NULL) {

if (this->tm->task->getStartTime() > startTime) {

TaskManagement\* temp = tm;

this->tm = newTm;

newTm->next = temp;

} else {

this->tm->next = newTm;

}

} else {

TaskManagement\* startEl = this->tm;

if (startEl->task->getStartTime() > startTime) {

newTm->next = startEl;

this->tm = newTm;

}

TaskManagement\* temp = this->tm;

TaskManagement\* temp1 = temp->next;;

while (temp1 != NULL ) {

if (startTime >= temp->task->getStartTime() && startTime <= temp1->task->getStartTime()) {

temp->next = newTm;

newTm->next = temp1;

break;

}

temp = temp->next;

temp1 = temp1->next;

}

if (temp1 == NULL) {

temp->next = newTm;

}

}

this->numTask += 1;

}

void output() {

cout <<"Ngay " << this->day << endl;

if (this->tm == NULL) cout << "No tasks";

else {

TaskManagement\* temp = this->tm;

while (temp != NULL) {

temp->task->printTask();

temp = temp->next;

}

}

}

int getAmountOfTask() {return this->numTask;}

int getDay() {return this->day;};

int compareDate(Date \*other) {

return this->day - other->getDay();

}

bool hasType(string type) {

if (this->tm == NULL) return false;

TaskManagement\* temp = this->tm;

int count = 0;

while (temp != NULL) {

if (temp->task->getType().compare(type) == 0) {

count++;

if (count == 1) cout << "Ngay " << this->day << endl;

temp->task->printTask();

}

temp = temp->next;

}

if (count > 0) return true;

return false;

}

bool hasStatus(string status) {

if (this->tm == NULL) return false;

TaskManagement\* temp = this->tm;

int count = 0;

while (temp != NULL) {

if (temp->task->getStatus().compare(status) == 0) {

count++;

if (count == 1) cout << "Ngay " << this->day << endl;

temp->task->printTask();

}

temp = temp->next;

}

if (count > 0) return true;

return false;

}

TaskManagement\* findTaskByName(string name) {

if (this->tm == NULL) return NULL;

TaskManagement\* temp = this->tm;

while (temp != NULL) {

if (temp->task->getName().compare(name) == 0) {

return temp;

}

temp = temp->next;

}

return NULL;

}

void replaceTaskByName(string oldName) {

if (findTaskByName(oldName)->task != NULL) {

string startTime, name, content, type, status;

cout << "Nhap cong viec(Gio bat dau(0-24)->ten->noi dung->tinh chat->trang thai): " << endl;

getline(cin, startTime);

getline(cin, name);

getline(cin, content);

getline(cin, type);

getline(cin, status);

Task\* newTask = new Task(startTime, name, content, type, status);

findTaskByName(oldName)->task = newTask;

if (this->status == '')

status = findTaskByName(oldName)->task->getStatus();

if (status.compare("Da thuc hien") == 0) {

removeElementByName(name);

}

} else {

cout << "Khong co cong viec nao";

}

cout << endl;

}

void removeElementByName(string name) {

if (this->tm == NULL) return;

TaskManagement\* temp = this->tm;

if (temp->task->getName().compare(name) == 0){

this->tm = NULL;

return ;

}

if (temp->next->task->getName().compare(name) == 0) {

temp->next = NULL;

return ;

}

TaskManagement\* temp2 = temp->next;

while(temp2->next != NULL) {

if (temp2->task->getName().compare(name) == 0) {

temp2->next = temp2->next->next;

}

temp2 = temp2->next;

}

if (temp2->task->getName().compare(name) == 0) temp2 = NULL;

}

};#endif

**3.3. Code class BST:**

#ifndef BST\_CPP

#define BST\_CPP

#include<bits/stdc++.h>

using namespace std;

class BST {

struct node {

Date\* data;

node\* left;

node\* right;

};

node\* root;

node\* makeEmpty(node\* t) {

if(t == NULL)

return NULL;

{

makeEmpty(t->left);

makeEmpty(t->right);

delete t;

}

return NULL;

}

node\* insert(Date\* x, node\* t)

{

if(t == NULL)

{

t = new node;

t->data = x;

t->left = t->right = NULL;

}

else if(x->compareDate(t->data) < 0)

t->left = insert(x, t->left);

else if(x->compareDate(t->data) > 0)

t->right = insert(x, t->right);

return t;

}

node\* findMin(node\* t)

{

if(t == NULL)

return NULL;

else if(t->left == NULL)

return t;

else

return findMin(t->left);

}

node\* findMax(node\* t) {

if(t == NULL)

return NULL;

else if(t->right == NULL)

return t;

else

return findMax(t->right);

}

node\* remove(Date\* x, node\* t) {

node\* temp;

if(t == NULL)

return NULL;

else if(x->compareDate(t->data) < 0)

t->left = remove(x, t->left);

else if(x->compareDate(t->data) > 0)

t->right = remove(x, t->right);

else if(t->left && t->right)

{

temp = findMin(t->right);

t->data = temp->data;

t->right = remove(t->data, t->right);

}

else

{

temp = t;

if(t->left == NULL)

t = t->right;

else if(t->right == NULL)

t = t->left;

delete temp;

}

return t;

}

void inorder(node\* t) {

if(t == NULL)

return;

inorder(t->left);

t->data->output();

inorder(t->right);

}

node\* find(node\* t, Date\* x) {

if(t == NULL)

return NULL;

else if(x->compareDate(t->data) < 0)

return find(t->left, x);

else if(x->compareDate(t->data) > 0)

return find(t->right, x);

else

return t;

}

void findByType(node\* t, string type) {

if(t == NULL) return;

else if (!t->data->hasType(type)) {

findByType(t->left, type);

findByType(t->right, type);

}

return ;

}

void findByStatus(node\* t, string status) {

if(t == NULL) return;

else if (!t->data->hasStatus(status)) {

findByStatus(t->left, status);

findByStatus(t->right, status);

}

return;

}

void findByDateRange(node \*t, int dayFrom, int dayTo) {

Date\* from = new Date(dayFrom);

Date\* to = new Date(dayTo);

if (t == NULL) return;

else if (from->compareDate(t->data) <= 0 && to->compareDate(t->data) >= 0) {

t->data->output();

} else {

findByDateRange(t->left, dayFrom, dayTo);

findByDateRange(t->right, dayFrom, dayTo);

}

}

public:

BST() {

root = NULL;

}

~BST() {

root = makeEmpty(root);

}

void insert(Date\* x) {

root = insert(x, root);

}

void remove(Date\* x) {

root = remove(x, root);

}

void display() {

inorder(root);

cout << endl;

}

void search(Date\* x) {

find(root, x)->data->output();

}

void searchByType(string type) {

findByType(root, type);

}

void searchByStatus(string status) {

findByStatus(root, status);

}

void searchByDateRange(int dayFrom, int dayTo) {

findByDateRange(root, dayFrom, dayTo);

}

void replaceTaskByName(string oldName) {

this->root->data->replaceTaskByName(oldName);

}

};#endif

**3.4. File Demo:**

#include "Task.cpp"

#include "Date.cpp"

#include "BST.cpp"

Date\* addTaskToDate(Date\* date) {

cout << "Nhap so cong viec: ";

int numTask; cin >> numTask;

cout << "Nhap cong viec(Gio bat dau(0-24)->ten->noi dung->tinh chat) " << endl;

string test;

getline(cin, test);

for (int i = 1; i <= numTask; i++) {

cout << "Nhap cong viec thu " << i << ":" << endl;

string startTime, content, name, type;

getline(cin, startTime);

getline(cin, name);

getline(cin, content);

getline(cin, type);

date->insert(startTime, name, content, type);

}

return date;

}

BST\* nhapNhiemVuMoi(BST\* t) {

cout << "Nhap so ngay can thuc hien: ";

int numDay; cin >> numDay;

for(int i = 1; i <= numDay; i++) {

int day;

cout << "Ngay thu " << i << " : "; cin >> day;

Date \* date = new Date(day);

t->insert(addTaskToDate(date));

}

return t;

}

void hienThiNhiemVuTheoNgay(BST \*t) {

cout << "Nhap ngay: ";

int dayCheck; cin >> dayCheck;

Date\* date = new Date(dayCheck);

t->search(date);

}

void hienThiNhiemVuTheoTinhChat(BST \*t) {

cout << "Nhap tinh chat cong viec can kiem tra: ";

string typeCheck;

getline(cin, typeCheck);

t->searchByType(typeCheck);

cout << "Tiep tuc? \n1.Yes\n2.No" << endl;

int option; cin >> option;

while(option == 1) {

cout << "Nhap tinh chat cong viec can kiem tra: ";

getline(cin, typeCheck);

getline(cin, typeCheck);

t->searchByType(typeCheck);

cout << "Tiep tuc? \n1.Yes\n2.No" << endl;

cin >> option;

}

}

void hienThiNhiemVuTheoTrangThai(BST \*t) {

cout << "Nhap trang thai cong viec can kiem tra: ";

string test;

getline(cin, test);

string statusCheck;

getline(cin, statusCheck);

t->searchByStatus(statusCheck);

cout << "Tiep tuc?\n1.Yes\n2.No" << endl;

int option; cin >> option;

while(option == 1) {

cout << "Nhap trang thai cong viec can kiem tra: ";

getline(cin, statusCheck);

t->searchByStatus(statusCheck);

cout << "Tiep tuc?\n1.Yes\n2.No" << endl;

cin >> option;

}

}

void hienThiNhiemVuTheoKhoangThoiGian(BST\* t) {

int from, to;

cout << "Nhap khoang gian(ngay): " << endl;

cout << "Tu: " ; cin >> from;

cout << "Den: "; cin >> to;

t->searchByDateRange(from, to);

}

void dieuChinhLichCongTac(BST\* t) {

t->display();

cout << endl;

cout << "Nhap ngay ";

int day; cin >> day;

Date\* date = new Date(day);

t->search(date);

string name;

cout << "Nhap ten cong viec muon dieu chinh ";

getline(cin, name);

getline(cin, name);

t->replaceTaskByName(name);

}

int main() {

BST\* t = new BST();

nhapNhiemVuMoi(t);

t->display();

return 0;

}

**IV. Phân tích thời gian chạy của từng phương thức có trong các lớp**

**4.1. Class Task:**

- Phương thức:

+ printTask(): O(1)

+ getContent(): O(1)

+ getType(): O(1)

+ getStatus(): O(1)

+ getName(): O(1)

+ getStartTime(): O(1)

**2.2. Class Date:**

- Phương thức:

+ insert(): O(n) (n là số nhiệm vụ hiện tại)

+ output(): O(1)

+ getAmountOfTask(): O(1)

+ getDay(): O(1)

+ compareDate(): O(1)

+ hasType(): O(n) (n là số nhiệm vụ hiện tại)

+ hasStatus(): O(n) (n là số nhiệm vụ hiện tại)

+ findTaskByName():O(n) (n là số nhiệm vụ hiện tại)

+ replaceTaskByName(): O(n) (n là số nhiệm vụ hiện tại)

+ removeElementByName(): O(n) (n là số nhiệm vụ hiện tại)

**2.3. Class BST:**

- Phương thức:

+ insert(): O(logn) n là số nhiệm vụ trong tháng

+ remove(): O(logn) n là số nhiệm vụ trong tháng

+ display(): O(1)

+ search(): O(logn) n là số nhiệm vụ trong tháng

+ searchByType(): O(logn) n là số nhiệm vụ trong tháng

+ searchByStatus(): O(logn) n là số nhiệm vụ trong tháng

+ searchByDateRange(): O(logn) n là số nhiệm vụ trong tháng

+ replaceTaskByName(): O(n) (n là số nhiệm vụ hiện tại)

**V. Tài liệu tham khảo**

- Bài làm có tham khảo bài giảng của giảng viên TS. Hoàng Văn Thông và trang web GeeksforGeeks.

https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/