My Project

Создано системой Doxygen 1.13.2

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Глава 2

Список файлов

2.1 Файлы

Полный список файлов.

Глава 3

Классы

3.1 Класс Employee

Класс, представляющий сотрудника.

#include <Employee.h>

Открытые члены

- Employee (const std::string &name, int hours_in_day)
  Конструктор сотрудника.
- void assign_task (Task *task)
  Назначить задачу сотруднику.
- void remove_task (Task *task)
  Удалить задачу у сотрудника.
- const std::string & get_name () const
  Получить имя сотрудника.
- int get_hours_in_day ()
  Получить доступные часы в день.
- const std::vector< Task * > & get_tasks () const
  Получить список задач.
- bool is_available () const
  Проверить доступность сотрудника.
- int total_assigned_hours () const
  Подсчитать общее количество часов всех задач.

Закрытые данные

- std::string _name
  Имя сотрудника
- int _hours_in_day
  Доступные часы в день
- std::vector< Task * > _tasks
  Список назначенных задач

3.1.1 Подробное описание

Класс, представляющий сотрудника.

3.1.2 Конструктор(ы)

3.1.2.1 Employee()

```
Employee::Employee (
            const std::string & name,
            int hours_in_day)
```

Конструктор сотрудника.

Аргументы

name	Имя сотрудника
hours_in_day	Количество доступных часов в день

3.1.3 Методы

3.1.3.1 assign_task()

```
void Employee::assign_task (
            Task * task)
```

Назначить задачу сотруднику.

Аргументы

task	Указатель на задачу

3.1.3.2 get_hours_in_day()

```
int Employee::get_hours_in_day ()
```

Получить доступные часы в день.

Возвращает

Количество часов

3.1.3.3 get_name()

const string & Employee::get_name () const

Получить имя сотрудника.

Возвращает

Ссылка на строку с именем

3.1.3.4 get_tasks()

const vector< Task * > & Employee::get_tasks () const

Получить список задач.

Возвращает

Вектор указателей на задачи

3.1.3.5 is_available()

bool Employee::is_available () const

Проверить доступность сотрудника.

Возвращает

true, если сотрудник доступен

3.1.3.6 remove_task()

void Employee::remove_task (
Task * task)

Удалить задачу у сотрудника.

Аргументы

task	Указатель на задачу

3.1.3.7 total_assigned_hours()

int Employee::total_assigned_hours () const

Подсчитать общее количество часов всех задач.

Возвращает

Сумма часов

3.1.4 Данные класса

3.1.4.1 _hours_in_day

int Employee::_hours_in_day [private]

Доступные часы в день

3.1.4.2 _name

std::string Employee::_name [private]

Имя сотрудника

3.1.4.3 _tasks

std::vector<Task*> Employee::_tasks [private]

Список назначенных задач

Объявления и описания членов классов находятся в файлах:

- Employee.h
- Employee.cpp

3.2 Класс Task

Класс, представляющий задачу.

#include <Task.h>

Открытые члены

- Task (std::string description, int workload_hours, int deadline_days, int status, Employee *assigned_employee)
  Конструктор с заданным ID.
- Task (int id, std::string description, int workload_hours, int deadline_days, int status, Employee *assigned_employee)
  Конструктор без ID (автоматически генерируется).
- int get_id () const
  Получить ID.
- std::string get_description () const
  Получить описание
- int & get_workload_hours ()
  Получить ссылку на трудоемкость
- int get_remaining_workload () const
  Получить оставшиеся часы
- void reduce_workload (int hours)

Уменьшает оставшуюся трудоёмкость.

- int & get_deadline_days ()
  Получить ссылку на дедлайн
- int get_status () const
  Получить статус
- void set_status (int status)
  Устанавливает статус задачи.
- Employee * get_assigned_employee ()
  Получить назначенного сотрудника
- void set_assigned_employee (Employee *employee)
  Назначить сотрудника

Открытые статические члены

- static void set_id_counter (int c)
  Установить текущий счетчик ID.

Закрытые данные

- int _id
  Уникальный идентификатор задачи
- std::string _description
  Описание задачи
- int _workload_hours
  Общая трудоемкость задачи
- int _remaining_workload
  Оставшиеся часы
- int _deadline_days
  Количество дней до дедлайна
- int _status
  Статус задачи
- Employee * _assigned_employee
  Назначенный сотрудник

Закрытые статические данные

- static int _id_counter = 0
  Глобальный счетчик ID.

3.2.1 Подробное описание

Класс, представляющий задачу.

3.2.2 Конструктор(ы)

3.2.2.1 Task() [1/2]

```
Task::Task (
            std::string description,
            int workload_hours,
            int deadline_days,
            int status,
            Employee * assigned_employee)
```

Конструктор с заданным ID.

Аргументы

id	Уникальный ID
description	Описание
workload_hours	Трудоемкость
deadline_days	Дедлайн
status	Статус
assigned_employee	Назначенный сотрудник

3.2.2.2 Task() [2/2]

```
Task::Task (
            int id,
            std::string description,
            int workload_hours,
            int deadline_days,
            int status,
            Employee * assigned_employee)
```

Конструктор без ID (автоматически генерируется).

3.2.3 Методы

3.2.3.1 get_assigned_employee()

```
Employee * Task::get_assigned_employee ()
```

Получить назначенного сотрудника

3.2.3.2 get_deadline_days()

```
int & Task::get_deadline_days ()
```

Получить ссылку на дедлайн

3.2.3.3 get_description()

```
std::string Task::get_description () const
```

Получить описание

3.2.3.4 get_id()

```
int Task::get_id () const
```

Получить ID.

3.2.3.5 get_remaining_workload()

int Task::get_remaining_workload () const

Получить оставшиеся часы

3.2.3.6 get_status()

int Task::get_status () const

Получить статус

3.2.3.7 get_workload_hours()

int & Task::get_workload_hours ()

Получить ссылку на трудоемкость

3.2.3.8 reduce_workload()

void Task::reduce_workload (
int hours)

Уменьшает оставшуюся трудоёмкость.

Аргументы

hours	Количество часов для вычитания.

3.2.3.9 set_assigned_employee()

void Task::set_assigned_employee (
Employee * employee)

Назначить сотрудника

3.2.3.10 set_id_counter()

void Task::set_id_counter (
int c) [static]

Установить текущий счетчик ID.

Аргументы

c	Новое значение счетчика

3.2.3.11 set_status()

void Task::set_status (
int status)

Устанавливает статус задачи.

Аргументы

status	Новый статус.

3.2.4 Данные класса

3.2.4.1 _assigned_employee

Employee* Task::_assigned_employee [private]

Назначенный сотрудник

3.2.4.2 _deadline_days

int Task::_deadline_days [private]

Количество дней до дедлайна

3.2.4.3 _description

std::string Task::_description [private]

Описание задачи

3.2.4.4 _id

int Task::_id [private]

Уникальный идентификатор задачи

3.2.4.5 _id_counter

int Task::_id_counter = 0 [static], [private]

Глобальный счетчик ID.

3.2.4.6 _remaining_workload

int Task::_remaining_workload [private]

Оставшиеся часы

3.2.4.7 _status

int Task::_status [private]

Статус задачи

3.2.4.8 _workload_hours

int Task::_workload_hours [private]

Общая трудоемкость задачи

Объявления и описания членов классов находятся в файлах:

- Task.h
- Task.cpp

3.3 Класс TaskManager

Класс управления задачами и сотрудниками.

#include <TaskManager.h>

Открытые члены

- TaskManager ()
  Конструктор по умолчанию.
- void save_file ()
  Сохраняет текущие данные в файл.
- void load_file ()
  Загружает данные из файла.
- const std::vector< Employee * > & get_all_employees ()
  Возвращает список всех сотрудников.
- const std::vector< Task * > & get_all_tasks ()
  Возвращает список всех задач.
- void add_employee (Employee *employee)
  Добавляет сотрудника в систему (список).
- void remove_employee (Employee *employee)
  Удаляет сотрудника из системы (списка).
- void search_employee (Employee *employee)
  Ищет сотрудника в системе (списке).
- void add_task (Task *task)
  Добавляет задачу в систему (список).
- void remove_task (Task *task)
  Удаляет задачу из системы (списка).
- void search_task (int id)
  Ищет задачу в системе (списке).
- void assign_task_to_employee (Task *task, Employee *employee)
  Назначает задачу сотруднику.
- void see_employment (Employee *employee)
  Показывает занятость конкретного сотрудника.
- void see_status ()
  Показывает статус всех задач.
- void task_ready ()
  Показывает статус всех задач.
- void end_day ()
  Завершает текущий день, обновляя дедлайны и прогресс задач.
- void timeskip ()
  Пропускает заданное количество дней.
- void show_all_employment ()
  Показывает занятость всех сотрудников.

Закрытые данные

- std::vector< Employee * > employees
  Список сотрудников
- std::vector< Task * > tasks
  Список задач
- int current_day
  Текущий день (для дедлайнов)

3.3.1 Подробное описание

Класс управления задачами и сотрудниками.

3.3.2 Конструктор(ы)

3.3.2.1 TaskManager()

TaskManager::TaskManager ()

Конструктор по умолчанию.

3.3.3 Методы

3.3.3.1 add_employee()

void TaskManager::add_employee (
Employee * employee)

Добавляет сотрудника в систему (список).

Аргументы

employee	Указатель на добавляемого сотрудника.

3.3.3.2 add_task()

void TaskManager::add_task (
Task * task)

Добавляет задачу в систему (список).

Аргументы

task	Указатель на добавляемую задачу.

3.3.3.3 assign_task_to_employee()

void TaskManager::assign_task_to_employee (
Task * task,
Employee * employee)

Назначает задачу сотруднику.

Аргументы

task	Указатель на задачу.
employee	Указатель на сотрудника.

3.3.3.4 end_day()

void TaskManager::end_day ()

Завершает текущий день, обновляя дедлайны и прогресс задач.

3.3.3.5 get_all_employees()

const std::vector< Employee * > & TaskManager::get_all_employees ()

Возвращает список всех сотрудников.

Возвращает

Ссылка на вектор сотрудников.

3.3.3.6 get_all_tasks()

const std::vector< Task * > & TaskManager::get_all_tasks ()

Возвращает список всех задач.

Возвращает

Ссылка на вектор сотрудников.

3.3.3.7 load_file()

void TaskManager::load_file ()

Загружает данные из файла.

3.3.3.8 remove_employee()

void TaskManager::remove_employee (
Employee * employee)

Удаляет сотрудника из системы (списка).

Аргументы

employee	Указатель на удаляемого сотрудника.

3.3.3.9 remove_task()

void TaskManager::remove_task (
Task * task)

Удаляет задачу из системы (списка).

Аргументы

task	Указатель на удаляемую задачу.

3.3.3.10 save_file()

void TaskManager::save_file ()

Сохраняет текущие данные в файл.

3.3.3.11 search_employee()

void TaskManager::search_employee (
Employee * employee)

Ищет сотрудника в системе (списке).

Аргументы

employee	Указатель на искомого сотрудника.

3.3.3.12 search_task()

void TaskManager::search_task (
int id)

Ищет задачу в системе (списке).

Аргументы

task	Указатель на искомую задачу.

3.3.3.13 see_employment()

void TaskManager::see_employment (
Employee * employee)

Показывает занятость конкретного сотрудника.

3.3.3.14 see_status()

void TaskManager::see_status ()

Показывает статус всех задач.

3.3.3.15 show_all_employment()

void TaskManager::show_all_employment ()

Показывает занятость всех сотрудников.

3.3.3.16 task_ready()

void TaskManager::task_ready ()

Показывает статус всех задач.

3.3.3.17 timeskip()

void TaskManager::timeskip ()

Пропускает заданное количество дней.

3.3.4 Данные класса

3.3.4.1 current_day

int TaskManager::current_day [private]

Текущий день (для дедлайнов)

3.3.4.2 employees

std::vector<Employee*> TaskManager::employees [private]

Список сотрудников

3.3.4.3 tasks

std::vector<Task*> TaskManager::tasks [private]

Список задач

Объявления и описания членов классов находятся в файлах:

- TaskManager.h
- TaskManager.cpp

Глава 4

Файлы

4.1 Файл Employee.cpp

```
#include "Employee.h"
#include "Task.h"
#include <iostream>
```

4.2 Файл Employee.h

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include "Task.h"
```

Классы

- class Employee

  Класс, представляющий сотрудника.

4.3 Employee.h

См. документацию.

```
00001 #pragma once
00002 #include <string>
00003 #include <iostream>
00004 #include <fstream>
00005 #include <vector>
00006 #include "Task.h"
00010
00011 class Employee {
00012 private:
00013     std::string _name;
00014     int _hours_in_day;
00015     std::vector<Task*> _tasks;
00016 public:
```

```
00022     Employee(const std::string& name, int hours_in_day);
00027     void assign_task(Task* task);
00032     void remove_task(Task* task);
00033
00038     const std::string& get_name() const;
00043     int get_hours_in_day();
00048     const std::vector<Task*>& get_tasks() const;
00049
00054     bool is_available() const;
00059     int total_assigned_hours() const;
00060 };
```

4.4 Файл main.cpp

```
#include <iostream>
#include <string>
#include "TaskManager.h"
```

Функции

- int main ()

4.4.1 Функции

4.4.1.1 main()

int main ()

4.5 Файл Task.cpp

```
#include "Task.h"
```

4.6 Файл Task.h

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
```

Классы

- class Task

  Класс, представляющий задачу.

4.7 Task.h

См. документацию.

```
00001 #pragma once
00002 #include <string>
00003 #include <iostream>
00004 #include <fstream>
00005 #include <vector>
00006 class Employee;
00007
00011
00012 class Task {
00013 private:
00014     static int _id_counter;
00015     int _id;
00016     std::string _description;
00017     int _workload_hours;
00018     int _remaining_workload;
00019     int _deadline_days;
00020     int _status;
00021     Employee* _assigned_employee;
00022 public:
00032     Task(std::string description, int workload_hours, int deadline_days, int status, Employee* assigned_employee);
00036     Task(int id, std::string description, int workload_hours, int deadline_days, int status, Employee* assigned_employee);
00037
00042     static void set_id_counter(int c);
00043
00044     int get_id() const;
00045     std::string get_description() const;
00046     int& get_workload_hours();
00047     int get_remaining_workload() const;
00052     void reduce_workload(int hours);
00053     int& get_deadline_days();
00054     int get_status() const;
00059     void set_status(int status);
00060     Employee* get_assigned_employee();
00061     void set_assigned_employee(Employee* employee);
00062 };
```

4.8 Файл TaskManager.cpp

```
#include "TaskManager.h"
```

4.9 Файл TaskManager.h

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include "Employee.h"
#include "Task.h"
```

Классы

- class TaskManager

  Класс управления задачами и сотрудниками.

4.10 TaskManager.h

См. документацию.

```
00001 #pragma once
00002 #include <string>
00003 #include <iostream>
00004 #include <fstream>
00005 #include <vector>
00006 #include "Employee.h"
00007 #include "Task.h"
00008
00012
00013 class TaskManager {
00014 private:
00015     std::vector<Employee*> employees;
00016     std::vector<Task*> tasks;
00017     int current_day;
00018 public:
00022     TaskManager();
00026     void save_file();
00030     void load_file();
00031
00036     const std::vector<Employee*>& get_all_employees();
00041     const std::vector<Task*>& get_all_tasks();
00042
00047     void add_employee(Employee* employee);
00052     void remove_employee(Employee* employee);
00057     void search_employee(Employee* employee);
00058
00063     void add_task(Task* task);
00068     void remove_task(Task* task);
00073     void search_task(int id);
00074
00080     void assign_task_to_employee(Task* task, Employee* employee);
00081
00085     void see_employment(Employee* employee);
00086
00090     void see_status();
00091
00095     void task_ready();
00096
00100     void end_day();
00104     void timeskip();
00105
00109     void show_all_employment();
00110 };
```

Предметный указатель