МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет | Автоматики и вычислительной техники |
| Кафедра | Автоматизированных систем управления |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Оценка комиссии: |  | | | Рейтинг: |  |
| Подписи членов комиссии: | | | | | |
|  | |  | Леонов Д.Г. | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | |  | Папилина Т.М. | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | | | | | |
| (дата) | | | | | |
|  | |  |  | | |

**КУРСОВАЯ РАБОТА**

|  |  |
| --- | --- |
| по дисциплине | Системное программирование |
|  | |

|  |  |
| --- | --- |
| на тему | Турнир роботов. Разработка клиентского модуля |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| «К ЗАЩИТЕ» |  | ВЫПОЛНИЛ: |  |
|  |  | Студент группы | **КИ-18-01** |
|  |  |  | (номер группы) |
|  |  | Поляков Александр Дмитриевич | |
| (должность, ученая степень; фамилия, и.о.) |  | (фамилия, имя, отчество) | |
|  |  |  | |
| (подпись) |  | (подпись) | |
|  |  |  | |
| (дата) |  | (дата) | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Москва, 20 | 21 |  |

МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет | Автоматики и вычислительной техник |
| Кафедра | Автоматизированных систем управления |

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

|  |  |
| --- | --- |
| по дисциплине | Системное программирование |
|  | |

|  |  |
| --- | --- |
| на тему | Турнир роботов. Разработка клиентского модуля |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| ДАНО студенту | Полякову А.Д. | группы | КИ-18-01 |
|  | (фамилия, имя, отчество в дательном падеже) |  | (номер группы) |

**Содержание работы:**

|  |  |
| --- | --- |
|  | Введение |
|  | Основная часть |
|  | Заключение |

**Исходные данные для выполнения работы:**

|  |  |
| --- | --- |
|  | Правила проведения турнира |
|  | Спецификации ядра |
|  |  |

**Рекомендуемая литература:**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Графическая часть:**

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель: |  |  |  |  |  |  |  |
|  | (уч.степень) |  | (должность) |  | (подпись) |  | (фамилия, имя, отчество) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Задание принял к исполнению: | студент |  |  |  | Поляков А.Д. |
|  | |  | (подпись) |  | (фамилия, имя, отчество) |

Оглавление

[Введение 4](#_Toc73290491)

[Правила турнира. 5](#_Toc73290492)

[Спецификации ядра. 6](#_Toc73290493)

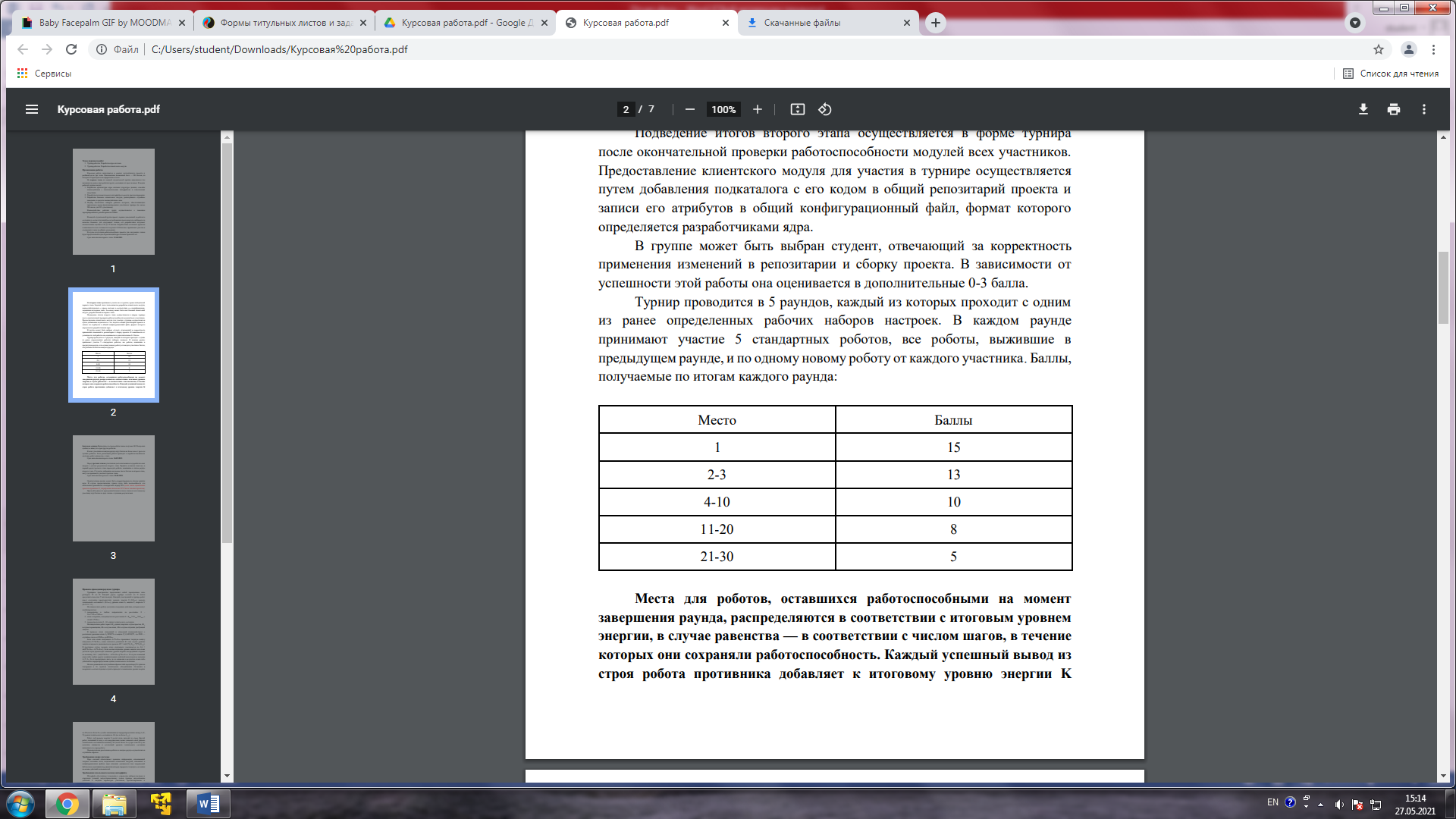
[Робот 8](#_Toc73290494)

# Введение

Разработка клиентского модуля, взаимодействующего с ядром системы в соответствии со спецификациями, заданными на первом этапе. За основу может быть взят базовый клиентский модуль, разработанный на первом этапе.

# Правила турнира.

Турнир проводится в 5 раундов, каждый из которых проходит с одним из ранее определенных рабочих наборов настроек. В каждом раунде принимают участие 5 стандартных роботов, все роботы, выжившие в предыдущем раунде, и по одному новому роботу от каждого участника. Баллы, получаемые по итогам каждого раунда:



Места для роботов, оставшихся работоспособными на момент

завершения раунда, распределяются в соответствии с итоговым уровнем

энергии, в случае равенства — в соответствии с числом шагов, в течение

которых они сохраняли работоспособность. Каждый успешный вывод из

строя робота противника добавляет к итоговому уровню энергии K

бонусных единиц. Вышедшие из строя роботы также получают K/2 бонусных

единиц за вывод из строя других роботов.

В зачет участника в каждом раунде идут баллы не более чем от трех его

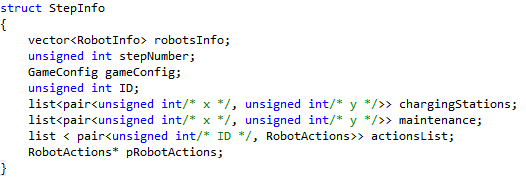
лучших роботов. Если реализация робота приводит к неработоспособности

системы, робот снимается с этапа.

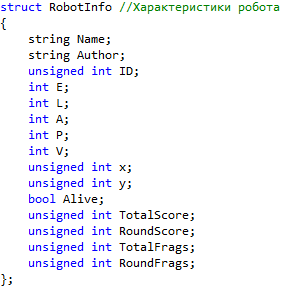
# Спецификации ядра.

Для добавления своего робота в общее решение, вам необходимо взять проект из папки Robots с именем RobotXX, где XX – номер по журналу (01, 02, …, 21). В основном cpp файле уже написана сигнатура функции DoStep с пустым телом. Все, что вам нужно сделать, это заполнить тело данной функции.

На каждом шаге роботу передается структура StepInfo:

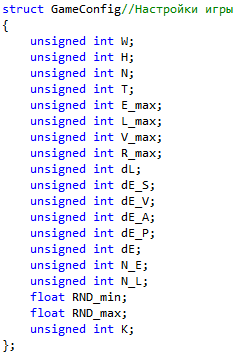


robotsInfo - информация обо всех роботах на текущем шаге, содержащая информацию следующего вида:



stepNumber – номер текущего шага;

gameConfig – настройки игры:



* ID – идентификационный номер робота
* chargingStations & maintenance – координаты станций подзарядки и технического обслуживания;
* actionsList – действия каждого робота на предыдущем шаге;
* pRobotActions – указатель на класс, хранящий действия вашего робота на текущем шаге;

# Робот

Функция, которая передаётся ядру игры:

extern "C" \_\_declspec(dllexport) void DoStep(StepInfo \* \_stepInfo)

{

srand(time(NULL));

stepInfo = \_stepInfo;

Чтение информации о роботе

for (auto it = stepInfo->robotsInfo.begin(); it != stepInfo->robotsInfo.end(); ++it)

{

if (\_stepInfo->ID == it->ID)

{

myInfo = new RobotInfo(\*it);

break;

}

}

Вычисление констант максимальной дистанции, максимальной дистанции атаки, максимальной силы атаки и максимальной защиты по формулам:

int Emax = stepInfo->gameConfig.E\_max;

int Lmax = stepInfo->gameConfig.L\_max;

int Vmax = stepInfo->gameConfig.V\_max;

int Rmax = stepInfo->gameConfig.R\_max;

int Smax = Vmax \* V / Lmax \* E / Emax;

int S\_amax = Rmax \* V / Lmax \* E / Emax;

int Amax = A \* E / Emax;

int Pmax = P \* E / Emax;

Объект для того чтобы узнавать расположение станций зарядки и ремонта:

Point energy, mech;

energy = getCStation(x, y);

mech = getMStation(x, y);

int se = Pythagoras(energy.x, energy.y);

int sm = Pythagoras(mech.x, mech.y);

Если у робота меньше 700 единиц энергии или осталось меньше 50 шагов, то он уходит на станцию подзарядки, распределяя все свои характеристики в атаку и защиту:

if ((se == 0) && ((E < 0.7 \* Emax) || (stepInfo->stepNumber > 950)) && (L > 0.7 \* Lmax))

{

stepInfo->pRobotActions->addActionRedistribution(0, L, 0);

return;

}

Если уровень энергии ниже критического, то робот уходит на станцию зарядки, перераспределяя характеристики.

if ((E < 0.7 \* Emax) || (stepInfo->stepNumber > 950))

{

if (Smax > se)

{

stepInfo->pRobotActions->addActionMove(energy.x, energy.y);

stepInfo->pRobotActions->addActionRedistribution(0, L, 0);

}

else

{

energy.x = energy.x \* Smax / se;

energy.y = energy.y \* Smax / se;

stepInfo->pRobotActions->addActionRedistribution(0, 0.4 \* L, 0.6 \* L);

stepInfo->pRobotActions->addActionMove(energy.x, energy.y);

}

return;

}

Информация о противниках и их местоположение:

VictimInfo vInfo = getVictimInfo();

if (vInfo.ID != 1000) {

RobotInfo\* victim;

for (auto it = stepInfo->robotsInfo.begin(); it != stepInfo->robotsInfo.end(); ++it)

{

if (vInfo.ID == it->ID)

{

victim = it.\_Ptr;

}

}

Если уровень энергии и технического состояния в пределах нормального, то считываем информацию о роботах и их местоположении:

VictimInfo getVictimInfo() {

UINT min\_dX = stepInfo->gameConfig.W + 1;

UINT min\_dY = stepInfo->gameConfig.H + 1;

int real\_dX = 0, real\_dY = 0;

UINT victimID = 1000;

for (auto it = stepInfo->robotsInfo.begin(); it != stepInfo->robotsInfo.end(); ++it)

Реализация союзов:

Point vec = getDistance(Point(myInfo->x, myInfo->y), Point(it->x, it->y));

if (Pythagoras(vec.x, vec.y) < Pythagoras(min\_dX, min\_dY) &&

it->Alive && myInfo->Author.compare(it->Author) != 0)

if (!(it->Author == "Yeah" || it->Author == "Lyaskin" || it->Author == "Polyakov"))

{

real\_dX = vec.x;

real\_dY = vec.y;

min\_dX = abs(vec.x);

min\_dY = abs(vec.y);

victimID = it->ID;

}

Рандомные значения для атаки и защиты:

float A\_r = stepInfo->gameConfig.RND\_min \* myInfo->A;

float P\_r = (1 - stepInfo->gameConfig.RND\_min) \* victim->P;

int delta = (A\_r \* myInfo->E - P\_r \* victim->E) / stepInfo->gameConfig.E\_max;

Логика атаки. Дистанция, расположение, расстояние и т.д.:

int maxDistToAttack = stepInfo->gameConfig.R\_max \* myInfo->V \* myInfo->E /

(stepInfo->gameConfig.L\_max \* stepInfo->gameConfig.E\_max);

int curDistance = Pythagoras(vInfo.dX, vInfo.dY);

if (curDistance <= maxDistToAttack && myInfo->E - stepInfo->gameConfig.dE\_A > 0)

{

stepInfo->pRobotActions->addActionAttack(vInfo.ID);

}

В конце задаём характеристики для робота, в сумме не превышающие 1:

stepInfo->pRobotActions->addActionRedistribution(0.5 \* L, 0.3 \* L, 0.2 \* L);

В переменную “direction” записывают значение движения, которые определяются рандомно. В зависимости от этого значения робот получает соответствующие команды

int direction = (rand() + myInfo->x) % 8;

switch (direction)

{

case 0:

stepInfo->pRobotActions->addActionMove(1, 0);

break;

case 1:

stepInfo->pRobotActions->addActionMove(-1, 0);

break;

case 2:

stepInfo->pRobotActions->addActionMove(0, 1);

break;

case 3:

stepInfo->pRobotActions->addActionMove(0, -1);

break;

case 4:

stepInfo->pRobotActions->addActionMove(1, 1);

break;

case 5:

stepInfo->pRobotActions->addActionMove(-1, 1);

break;

case 6:

stepInfo->pRobotActions->addActionMove(-1, -1);

break;

case 7:

stepInfo->pRobotActions->addActionMove(1, -1);

break;

default:

break;

}

return;

}

}