Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по творческой работе**

Дисциплина: Информатика

Тема: Разработка калькулятора и решение задачи Комивояжера.

Выполнил работу

студент группы РИС-20-1б

Ремянников А. В.

Проверила

Доцент кафедры ИТАС

к.т.н Полякова О. А.

Пермь, 2021

## Первая часть – Калькулятор

**Постановка задачи**

(Вариант 19) Необходимо создать турнирную таблицу, в которой вводятся названия команд, результаты матчей, считается количество выигранных и проигранных матчей, количество ничьих, количество набранных очков. По окончании турнира команды должны быть расположены в соответствие с набранными очками от максимума к минимуму. На панели калькулятора отображаются исходные данные и результаты, операции и служебные клавиши.

**Анализ задачи**

**Предварительный анализ**

Поскольку футбольные турниры могут проводиться по различным системам, было принято решение расширить задачу до реализации трех различных турнирных систем: Олимпийской, С выбыванием после двух поражений и Круговой.

Язык реализации программы - C++, как один из наиболее актуальных и многофункциональных на данный момент. Для реализации пользовательского интерфейса было принято решение использовать фреймворк QT, так как он предоставляет удобные инструменты для работы с интерфейсом. Для создания и тестирования программы использовалась интегрированная среда разработки QT Creator.

**Анализ предметной области**

Перед реализацией программы была проанализирована предметная область, в целях улучшения качества конечного продукта.

Олимпийская система в спортивных соревнованиях - система, при которой участник выбывает из турнира после первого же проигрыша, а победителем называется команда, не проигравшая ни одного матча. Так, олимпийская система обеспечивает выявление победителя за минимальное число туров.

Круговая система, система, при которой каждый участник турнира играет с каждым, победитель – набравший наибольшее количество очков. За победу команде- победительнице начисляется 2 очка, при ничье обе команды получают по одному очку.

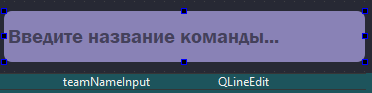
Турнирная система с выбыванием после двух поражений - система проведения турниров, в которой участник выбывает из турнира после двух поражений.

Так как для олимпийской системы и системы в выбыванием после двух поражений необходимым условием является то, что количество команд должно быть степенью двойки, то при количестве команд неудовлетворяющем этому условию проведение турнира по таким системам невозможно. Поэтому в спортивных соревнованиях предусмотрены предварительные раунды, которые проводятся по круговой системе, с целью сократить количество команд-участниц. Поэтому будет необходимо реализовать возможность проведения предварительных этапов.

**Ход работы**

**1. Создание класса главного окна, позволяющего определить список команд-участниц предстоящего турнира.**

**1.1**. Для ввода названий команд использовался объект интерфейса QLineEdit, которой представляет собой строку ввода данных.



Данные, полученные с помощью данного элемента интерфейса представляются в специальном типе QString, поэтому для удобного их хранения и обработки требовалось преобразовать их в стандартный тип string, для этого использовался метод класса QString - toUtf8().

ui->teamNameInput->text().toUtf8().constData();

**1.2.** Для хранения названий команд использовался массив из элементов стандартного типа string.

std::string \*teamNameList=new std::string[100];

Для добавления названий команд в массив была реализована функция addTeamNameInList().

void CalcClass::addTeamNameInList()

{

if (strcmp(ui->teamNameInput->text().toUtf8().constData(), "") != 0)

{

teamNameList[row-1]=ui->teamNameInput->text().toUtf8().constData();

ui->teamNameInput->clear();

}

}

**1.3.** Для отображения добавленных имен команд использовался объект класса QTableWidget, работающий как обычная таблица, в которую можно заносить элементы.



Для добавления элементов в таблицу была реализована функция

void CalcClass::addTeamNameInTable()

{

if (strcmp( ui->teamNameInput->text().toUtf8().constData(), "") != 0)

{

ui->teamNameTable->setRowCount(row+1);

QTableWidgetItem \*tbl= new QTableWidgetItem(ui->teamNameInput->text());

ui->teamNameTable->setItem(row,0,tbl);

row++;

}

}

**1.4.** Для удобства и универсальности использования программы была реализована функция, позволяющая заполнить список команд названиями из файла пользователя.

void CalcClass::fillTableFromFile()

{

if (fileNotChecked)

{

std::ifstream teamNames("C:/Users/Sasha/Documents/QtProjects/Calculator/teamNames.txt");

if (teamNames.is\_open())

{

while (!teamNames.eof())

{

std::string teamName;

getline(teamNames,teamName);

if (teamName!="\n" && teamName!="")

{

ui->teamNameTable->setRowCount(row+1);

teamNameList[row]=teamName;

QTableWidgetItem \*tbl= new QTableWidgetItem(QString::fromUtf8(teamNameList[row].c\_str()));

ui->teamNameTable->setItem(row,0,tbl);

row++;

}

}

}

fileNotChecked=false;

}

}

**1.5.** Была реализована кнопка, закрывающая главное окно и открывающая окна для различных турнирных систем, для этого была реализована функция

void CalcClass::teamNameTableFinished()

{

if (row>1)

{

if (ui->checkBoxOlymp->isChecked())

{

close();

if (row==2 || row==4 || row==8 || row ==16)

{

olympWindowClass \*olympWindow=new olympWindowClass;

olympWindow->send(teamNameList,row);

olympWindow->move(10,200);

olympWindow->show();

}

else

{

roundRobinWindowClass \*roundRobinWindow= new roundRobinWindowClass;

roundRobinWindow->send(teamNameList, row, true, false);

roundRobinWindow->move(0,210);

roundRobinWindow->show();

}

}

if (ui->checkBoxRoundRobin->isChecked())

{

close();

roundRobinWindowClass \*roundRobinWindow= new roundRobinWindowClass;

roundRobinWindow->send(teamNameList, row, false, false);

roundRobinWindow->move(850,210);

roundRobinWindow->show();

}

if (ui->checkBoxDoubleElimination->isChecked())

{

if (row==2 || row==4 || row==8 || row ==16)

{

close();

doubleEliminationWindowClass \*doubleEliminationWindow= new doubleEliminationWindowClass;

doubleEliminationWindow->send(teamNameList, row);

doubleEliminationWindow->move(340,0);

doubleEliminationWindow->show();

}

else

{

roundRobinWindowClass \*roundRobinWindow= new roundRobinWindowClass;

roundRobinWindow->send(teamNameList, row, false, true);

roundRobinWindow->move(400,210);

roundRobinWindow->show();

}

}

}

}

**2. Создание классов для дополнительных окон различных турнирных систем.**

**2.1.** Реализация класса для окна олимпийской системы.

**2.1.1.** Была реализована функция для передачи в класс олимпийской системы массива и его размера.

void olympWindowClass::send(std::string\* mas, int n)

{

teamNameList=mas;

length=n;

for (int i=0; i<length; i++)

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

labelForTeamName->setMinimumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

labelForTeamName->setText(QString::number(i%2+1)+' '+QString::fromUtf8(teamNameList[i].c\_str()));

ui->nameLayout1->addWidget(labelForTeamName);

nameLayoutNumber=2;

}

close();

}

**2.1.2.** Была реализована функция для обработки введенного результата матча, которая динамически добавляет объекты QLabel для отображения названия команд.

void olympWindowClass::resultEntered()

{

if (length>1)

{

switch(nameLayoutNumber)

{

case 2:

if (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

for (int i=k+1; i<length-1; i++) teamNameList[i]=teamNameList[i+1];

length--;

if (length==1) labelForTeamName->setText(QString::fromUtf8("Winner: ")+' '+QString::fromUtf8(teamNameList[k].c\_str()));

else labelForTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(teamNameList[k].c\_str()));

ui->nameLayout2->addWidget(labelForTeamName);

k++;

ui->resultLine->clear();

}

if (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

for (int i=k; i<length-1; i++) teamNameList[i]=teamNameList[i+1];

length--;

if (length==1) labelForTeamName->setText(QString::fromUtf8("Winner: ")+' '+QString::fromUtf8(teamNameList[k].c\_str()));

else labelForTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(teamNameList[k].c\_str()));

ui->nameLayout2->addWidget(labelForTeamName);

k++;

ui->resultLine->clear();

}

if (k==length)

{

nameLayoutNumber++;

k=0;

}

break;

case 3:

if (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

for (int i=k+1; i<length-1; i++) teamNameList[i]=teamNameList[i+1];

length--;

if (length==1) labelForTeamName->setText(QString::fromUtf8("Winner: ")+' '+QString::fromUtf8(teamNameList[k].c\_str()));

else labelForTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(teamNameList[k].c\_str()));

ui->nameLayout3->addWidget(labelForTeamName);

k++;

ui->resultLine->clear();

}

if (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

for (int i=k; i<length-1; i++) teamNameList[i]=teamNameList[i+1];

length--;

if (length==1) labelForTeamName->setText(QString::fromUtf8("Winner: ")+' '+QString::fromUtf8(teamNameList[k].c\_str()));

else labelForTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(teamNameList[k].c\_str()));

ui->nameLayout3->addWidget(labelForTeamName);

k++;

ui->resultLine->clear();

}

if (k==length)

{

nameLayoutNumber++;

k=0;

}

break;

case 4:

if (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

for (int i=k+1; i<length-1; i++) teamNameList[i]=teamNameList[i+1];

length--;

if (length==1) labelForTeamName->setText(QString::fromUtf8("Winner: ")+' '+QString::fromUtf8(teamNameList[k].c\_str()));

else labelForTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(teamNameList[k].c\_str()));;

ui->nameLayout4->addWidget(labelForTeamName);

k++;

ui->resultLine->clear();

}

if (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

for (int i=k; i<length-1; i++) teamNameList[i]=teamNameList[i+1];

length--;

if (length==1) labelForTeamName->setText(QString::fromUtf8("Winner: ")+' '+QString::fromUtf8(teamNameList[k].c\_str()));

else labelForTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(teamNameList[k].c\_str()));

ui->nameLayout4->addWidget(labelForTeamName);

k++;

ui->resultLine->clear();

}

if (k==length)

{

nameLayoutNumber++;

k=0;

}

break;

case 5:

if (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

for (int i=k+1; i<length-1; i++) teamNameList[i]=teamNameList[i+1];

length--;

if (length==1) labelForTeamName->setText(QString::fromUtf8("Winner: ")+' '+QString::fromUtf8(teamNameList[k].c\_str()));

else labelForTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(teamNameList[k].c\_str()));

ui->nameLayout5->addWidget(labelForTeamName);

k++;

ui->resultLine->clear();

}

if (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 12);

labelForTeamName->setFont(bold);

for (int i=k; i<length-1; i++) teamNameList[i]=teamNameList[i+1];

length--;

if (length==1) labelForTeamName->setText(QString::fromUtf8("Winner: ")+' '+QString::fromUtf8(teamNameList[k].c\_str()));

else labelForTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(teamNameList[k].c\_str()));

ui->nameLayout5->addWidget(labelForTeamName);

k++;

ui->resultLine->clear();

}

if (k==length)

{

nameLayoutNumber++;

k=0;

}

break;

}

}

ui->resultLine->clear();

}

**2.1.3.** Был реализован пользовательский интерфейс для окна олимпийской системы.



**2.2.** Реализация класса для окна круговой системы.

**2.2.1.** Была реализована функция для передачи в класс круговой системы массива и его размера, а также флажков для того, чтобы определить- используется ли круговая система как предварительный этап для олимпийской системы или системы с выбыванием после двух поражений.

void roundRobinWindowClass::send(std::string\* mas, int n, bool f1, bool f2)

{

teamNameList=mas;

length=n;

forPreOlympNeeds=f1;

forPreDoubleNeeds=f2;

for (int y=0; y<length; y++) points[y]=0;

if (forPreOlympNeeds||forPreDoubleNeeds)

{

QDynamicLabel \*preOlympMessage = new QDynamicLabel(this);

QFont bold("Franklin Gothic Demi", 10);

preOlympMessage->setFont(bold);

preOlympMessage->setText("Количество команд не является степенью двойки или больше 16, необходим предварительный этап (по круговой системе)");

ui->preOlympLayout->addWidget(preOlympMessage);

connect(ui->finishButton, &QPushButton::pressed, this, &roundRobinWindowClass::preOlympStageFinished);

ui->finishButton->setText("Завершить предварительный этап");

}

else

{

ui->finishButton->setText("Завершить турнир");

connect(ui->finishButton, &QPushButton::pressed, this, &roundRobinWindowClass::closeWindow);

}

ui->nameLabel1->setText(QString::fromUtf8(teamNameList[i].c\_str()));

ui->nameLabel2->setText(QString::fromUtf8(teamNameList[j].c\_str()));

int c=0;

ui->resultTable->setRowCount(length);

while (c<length)

{

QTableWidgetItem \*tbl1= new QTableWidgetItem(QString::fromUtf8(teamNameList[c].c\_str()));

ui->resultTable->setItem(c,0,tbl1);

QTableWidgetItem \*tbl2= new QTableWidgetItem(QString::number(0));

ui->resultTable->setItem(c,1,tbl2);

c++;

}

}

**2.2.2.** Была реализована функция для обработки введенного результата матча, которая начисляет баллы победившей команде в специальный массив, а также вызывает функцию обновления турнирной таблицы.

void roundRobinWindowClass::resultEntered()

{

if (i<length && j<length)

{

if (goals1==-1)

{

goals1=ui->resultLine->text().toInt();

ui->resultLabel->setText(QString::number(goals1)+" : ?");

}

else

{

if (goals2==-1)

{

goals2=ui->resultLine->text().toInt();

ui->resultLabel->setText(QString::number(goals1)+" : "+QString::number(goals2));

}

}

if (goals1!=-1 && goals2!=-1)

{

if (goals1>goals2)

{

points[i]+=2;

}

if (goals1<goals2)

{

points[j]+=2;

}

if (goals1==goals2)

{

points[i]+=1;

points[j]+=1;

}

goals1=-1;

goals2=-1;

ui->resultLabel->setText("? : ?");

this->updateResultTable();

}

}

ui->resultLine->clear();

}

**2.2.3.** Была реализована функция обновления турнирной таблицы интерфейса.

void roundRobinWindowClass::updateResultTable()

{

QTableWidgetItem \*tbl1= new QTableWidgetItem(QString::number(points[i]));

ui->resultTable->setItem(i,1,tbl1);

QTableWidgetItem \*tbl2= new QTableWidgetItem(QString::number(points[j]));

ui->resultTable->setItem(j,1,tbl2);

if (j<length-1)

{

j++;

}

else

{

i++;

j=i+1;

}

ui->nameLabel1->setText(QString::fromUtf8(teamNameList[i].c\_str()));

ui->nameLabel2->setText(QString::fromUtf8(teamNameList[j].c\_str()));

if (i==length || j==length)

{

int max=-1;

int maxpos=0;

std::string temp="";

for (int x=0; x<length; x++)

{

max=-1;

for (int y=x; y<length; y++)

{

if (points[y]>max)

{

max=points[y];

maxpos=y;

}

}

points[maxpos]=points[x];

points[x]=max;

temp=teamNameList[maxpos];

teamNameList[maxpos]=teamNameList[x];

teamNameList[x]=temp;

}

for (int z=0; z<length; z++)

{

QTableWidgetItem \*tbl1= new QTableWidgetItem(QString::number(points[z]));

ui->resultTable->setItem(z,1,tbl1);

QTableWidgetItem \*tbl2= new QTableWidgetItem(QString::fromUtf8(teamNameList[z].c\_str()));

ui->resultTable->setItem(z,0,tbl2);

}

ui->resultLabel->setText("Все матчи проведены");

ui->nameLabel1->setText("");

ui->nameLabel2->setText("");

}

}

**2.2.4.** Была реализована функция, срабатывающая при нажатии кнопки завершения турнира, в случае, если флажки, переданные в методе send сообщают о том, что круговая система использовалась как предварительный этап. Функция открывает окно олимпийской системы или системы с выбыванием послу двух поражений и передает в эти окна обновленный массив имен команд.

void roundRobinWindowClass::preOlympStageFinished()

{

if (forPreOlympNeeds && notOpened)

{

close();

olympWindowClass \*olympWindow=new olympWindowClass;

std::string\* temp=new std::string[100];

if (length>16) length=16;

if (length>8 && length<16) length=8;

if (length>4 && length<8) length=4;

if (length>2 && length<4) length=2;

for (int x=0; x<length; x++) temp[x]=teamNameList[x];

olympWindow->send (temp, length);

olympWindow->move(10,200);

olympWindow->show();

notOpened=false;

}

if (forPreDoubleNeeds && notOpened)

{

close();

doubleEliminationWindowClass \*doubleEliminationWindow= new doubleEliminationWindowClass;

std::string\* temp=new std::string[100];

if (length>16) length=16;

if (length>8 && length<16) length=8;

if (length>4 && length<8) length=4;

if (length>2 && length<4) length=2;

for (int x=0; x<length; x++) temp[x]=teamNameList[x];

doubleEliminationWindow->send (temp, length);

doubleEliminationWindow->move(350,0);

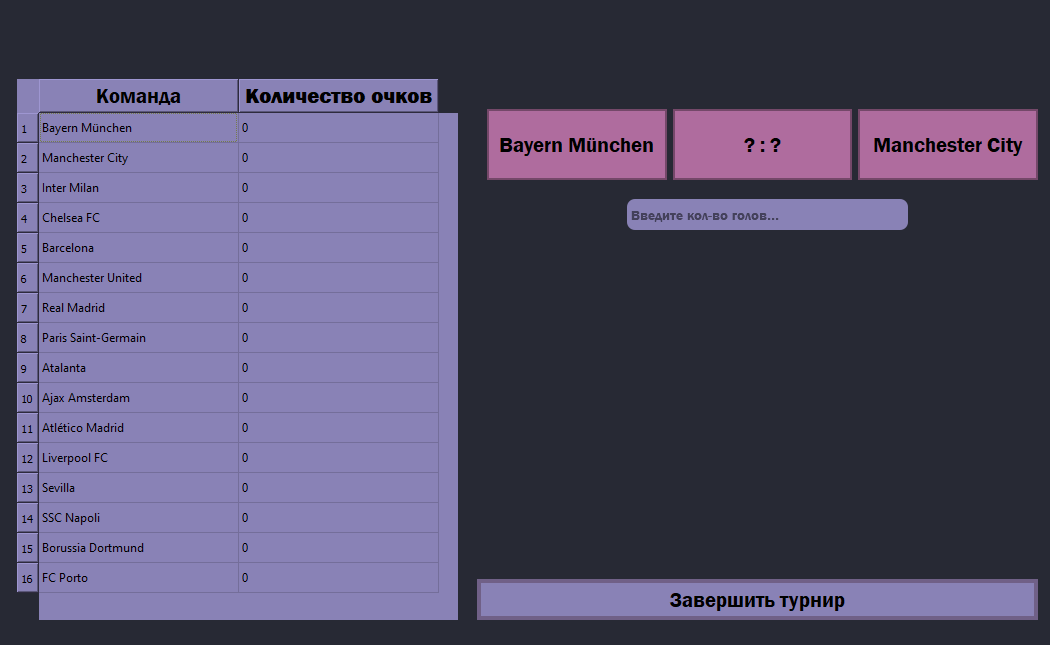
doubleEliminationWindow->show();

notOpened=false;

}

}

**2.2.5.** Был реализован пользовательский интерфейс для окна круговой системы.



**2.3.** Реализация класса для окна системы с выбыванием после двух поражений.

**2.3.1.** Была реализована функция для передачи в класс системы с выбыванием после двух поражений массива имен и его размера.

void doubleEliminationWindowClass::send(std::string\* mas, int n)

{

uteamNameList=mas;

ulength=n;

llength=0;

if (n==2)

{

only2teams=true;

fteamNameList=mas;

flength=2;

QDynamicLabel \*labelFor1TeamName = new QDynamicLabel(this);

QDynamicLabel \*labelFor2TeamName = new QDynamicLabel(this);

QFont bold("Franklin Gothic Demi", 16);

labelFor1TeamName->setMaximumHeight(40);

labelFor2TeamName->setMaximumHeight(40);

labelFor1TeamName->setFont(bold);

labelFor2TeamName->setFont(bold);

labelFor1TeamName->setText(QString::number(1)+' '+QString::fromUtf8(fteamNameList[0].c\_str()));

ui->final\_2->addWidget(labelFor1TeamName);

labelFor2TeamName->setText(QString::number(2)+' '+QString::fromUtf8(fteamNameList[1].c\_str()));

ui->final\_2->addWidget(labelFor2TeamName);

}

else

{

for (int i=0; i<ulength; i++)

{

QDynamicLabel \*labelForTeamName = new QDynamicLabel(this);

labelForTeamName->setMaximumHeight(30);

labelForTeamName->setMinimumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

labelForTeamName->setText(QString::number(i%2+1)+' '+QString::fromUtf8(uteamNameList[i].c\_str()));

ui->upperRound1->addWidget(labelForTeamName);

round=1;

}

}

close();

}

**2.3.2.** Была реализована функция, обрабатывающая результаты матчей и динамически добавляющая элементы класса QLabel для отображения названий команд.

void doubleEliminationWindowClass::**resultEntered**()

{

*if* ((flength==2 || only2teams) && (notEnded))

{

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(40);

QFont bold("Franklin Gothic Demi", 16);

labelForTeamName->setFont(bold);

labelForTeamName->setText("winner: "+QString::fromUtf8(fteamNameList[0].c\_str()));

ui->winner->addWidget(labelForTeamName);

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(40);

QFont bold("Franklin Gothic Demi", 16);

labelForTeamName->setFont(bold);

labelForTeamName->setText("winner: "+QString::fromUtf8(lteamNameList[1].c\_str()));

ui->winner->addWidget(labelForTeamName);

ui->resultLine->clear();

}

notEnded=*false*;

}

*if* ((ulength>1 || llength>1) && !(only2teams))

{

*if* (upperBracket)

{

*switch* (round)

{

*case* 1:

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForWTeamName = *new* QDynamicLabel(*this*);

labelForWTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForWTeamName->setFont(bold);

labelForWTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

*if* (ulength>2)

{

ui->upperRound2->addWidget(labelForWTeamName);

}

*else*

{

labelForWTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

ui->final\_2->addWidget(labelForWTeamName);

fteamNameList[flength]=uteamNameList[k];

flength++;

}

*if* (ulength>2)

{

QDynamicLabel \*labelForLTeamName = *new* QDynamicLabel(*this*);

labelForLTeamName->setMaximumHeight(30);

labelForLTeamName->setFont(bold);

labelForLTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

ui->lowerRound1->addWidget(labelForLTeamName);

lteamNameList[llength]=uteamNameList[k+1];

llength++;

}

*for* (int i=k+1; i<ulength-1; i++) uteamNameList[i]=uteamNameList[i+1];

ulength--;

k++;

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForWTeamName = *new* QDynamicLabel(*this*);

labelForWTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForWTeamName->setFont(bold);

labelForWTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

*if* (ulength>2)

{

ui->upperRound2->addWidget(labelForWTeamName);

}

*else*

{

labelForWTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

ui->final\_2->addWidget(labelForWTeamName);

fteamNameList[flength]=uteamNameList[k+1];

flength++;

}

*if* (ulength>2)

{

QDynamicLabel \*labelForLTeamName = *new* QDynamicLabel(*this*);

labelForLTeamName->setMaximumHeight(30);

labelForLTeamName->setFont(bold);

labelForLTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

ui->lowerRound1->addWidget(labelForLTeamName);

lteamNameList[llength]=uteamNameList[k];

llength++;

}

*for* (int i=k; i<ulength-1; i++) uteamNameList[i]=uteamNameList[i+1];

ulength--;

k++;

ui->resultLine->clear();

}

*if* (k==ulength)

{

upperBracket=*false*;

k=0;

}

*break*;

*case* 2:

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForWTeamName = *new* QDynamicLabel(*this*);

labelForWTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForWTeamName->setFont(bold);

labelForWTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

*if* (ulength>2)

{

ui->upperRound3->addWidget(labelForWTeamName);

}

*else*

{

labelForWTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

ui->final\_2->addWidget(labelForWTeamName);

fteamNameList[flength]=uteamNameList[k];

flength++;

}

*if* (ulength>2)

{

QDynamicLabel \*labelForLTeamName = *new* QDynamicLabel(*this*);

labelForLTeamName->setMaximumHeight(30);

labelForLTeamName->setFont(bold);

labelForLTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

ui->lowerRound2->addWidget(labelForLTeamName);

lteamNameList[llength]=uteamNameList[k+1];

llength++;

}

*for* (int i=k+1; i<ulength-1; i++) uteamNameList[i]=uteamNameList[i+1];

ulength--;

k++;

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForWTeamName = *new* QDynamicLabel(*this*);

labelForWTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForWTeamName->setFont(bold);

labelForWTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

*if* (ulength>2)

{

ui->upperRound3->addWidget(labelForWTeamName);

}

*else*

{

labelForWTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

ui->final\_2->addWidget(labelForWTeamName);

fteamNameList[flength]=uteamNameList[k+1];

flength++;

}

*if* (ulength>2)

{

QDynamicLabel \*labelForLTeamName = *new* QDynamicLabel(*this*);

labelForLTeamName->setMaximumHeight(30);

labelForLTeamName->setFont(bold);

labelForLTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

ui->lowerRound2->addWidget(labelForLTeamName);

lteamNameList[llength]=uteamNameList[k];

llength++;

}

*for* (int i=k; i<ulength-1; i++) uteamNameList[i]=uteamNameList[i+1];

ulength--;

k++;

ui->resultLine->clear();

}

*if* (k==ulength)

{

upperBracket=*false*;

k=0;

}

*break*;

*case* 3:

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForWTeamName = *new* QDynamicLabel(*this*);

labelForWTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForWTeamName->setFont(bold);

labelForWTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

*if* (ulength>2)

{

ui->upperRound4->addWidget(labelForWTeamName);

}

*else*

{

labelForWTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

ui->final\_2->addWidget(labelForWTeamName);

fteamNameList[flength]=uteamNameList[k];

flength++;

}

*for* (int i=k+1; i<ulength-1; i++) uteamNameList[i]=uteamNameList[i+1];

ulength--;

k++;

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForWTeamName = *new* QDynamicLabel(*this*);

labelForWTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForWTeamName->setFont(bold);

labelForWTeamName->setText(QString::number(k%2+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

*if* (ulength>2)

{

ui->upperRound4->addWidget(labelForWTeamName);

}

*else*

{

labelForWTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

ui->final\_2->addWidget(labelForWTeamName);

fteamNameList[flength]=uteamNameList[k+1];

flength++;

}

*for* (int i=k; i<ulength-1; i++) uteamNameList[i]=uteamNameList[i+1];

ulength--;

k++;

ui->resultLine->clear();

}

*if* (k==ulength)

{

upperBracket=*false*;

k=0;

}

*break*;

*case* 4:

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForWTeamName = *new* QDynamicLabel(*this*);

labelForWTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForWTeamName->setFont(bold);

labelForWTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(uteamNameList[k].c\_str()));

ui->final\_2->addWidget(labelForWTeamName);

fteamNameList[flength]=uteamNameList[k];

flength++;

*for* (int i=k+1; i<ulength-1; i++) uteamNameList[i]=uteamNameList[i+1];

ulength--;

k++;

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForWTeamName = *new* QDynamicLabel(*this*);

labelForWTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForWTeamName->setFont(bold);

labelForWTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(uteamNameList[k+1].c\_str()));

ui->final\_2->addWidget(labelForWTeamName);

fteamNameList[flength]=uteamNameList[k+1];

flength++;

*for* (int i=k; i<ulength-1; i++) uteamNameList[i]=uteamNameList[i+1];

ulength--;

k++;

ui->resultLine->clear();

}

*if* (k==ulength)

{

upperBracket=*false*;

k=0;

}

*break*;

}

}

*else*

{

*switch*(round)

{

*case* 1:

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

*for* (int i=l+1; i<llength-1; i++) lteamNameList[i]=lteamNameList[i+1];

llength--;

labelForTeamName->setText(QString::number(l%2+1)+' '+QString::fromUtf8(lteamNameList[l].c\_str()));

*if* (llength>1) ui->lowerRound2->addWidget(labelForTeamName);

*else*

{

labelForTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(lteamNameList[l].c\_str()));

ui->final\_2->addWidget(labelForTeamName);

fteamNameList[flength]=lteamNameList[l];

flength++;

}

l++;

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

*for* (int i=l; i<llength-1; i++) lteamNameList[i]=lteamNameList[i+1];

llength--;

labelForTeamName->setText(QString::number(l%2+1)+' '+QString::fromUtf8(lteamNameList[l+1].c\_str()));

*if* (llength>1) ui->lowerRound2->addWidget(labelForTeamName);

*else*

{

labelForTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(lteamNameList[l+1].c\_str()));

ui->final\_2->addWidget(labelForTeamName);

fteamNameList[flength]=lteamNameList[l+1];

flength++;

}

l++;

ui->resultLine->clear();

}

*if* (l==llength)

{

round++;

upperBracket=*true*;

l=0;

}

*break*;

*case* 2:

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

*for* (int i=l+1; i<llength-1; i++) lteamNameList[i]=lteamNameList[i+1];

llength--;

labelForTeamName->setText(QString::number(l%2+1)+' '+QString::fromUtf8(lteamNameList[l].c\_str()));

*if* (llength>1) ui->lowerRound3->addWidget(labelForTeamName);

*else*

{

labelForTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(lteamNameList[l].c\_str()));

ui->final\_2->addWidget(labelForTeamName);

fteamNameList[flength]=lteamNameList[l];

flength++;

}

l++;

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

*for* (int i=l; i<llength-1; i++) lteamNameList[i]=lteamNameList[i+1];

llength--;

labelForTeamName->setText(QString::number(l%2+1)+' '+QString::fromUtf8(lteamNameList[l+1].c\_str()));

*if* (llength>1) ui->lowerRound3->addWidget(labelForTeamName);

*else*

{

labelForTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(lteamNameList[l+1].c\_str()));

ui->final\_2->addWidget(labelForTeamName);

fteamNameList[flength]=lteamNameList[l+1];

flength++;

}

l++;

ui->resultLine->clear();

}

*if* (l==llength)

{

round++;

upperBracket=*true*;

l=0;

}

*break*;

*case* 3:

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

*for* (int i=l+1; i<llength-1; i++) lteamNameList[i]=lteamNameList[i+1];

llength--;

labelForTeamName->setText(QString::number(l%2+1)+' '+QString::fromUtf8(lteamNameList[l].c\_str()));

*if* (llength>1) ui->lowerRound4->addWidget(labelForTeamName);

*else*

{

labelForTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(lteamNameList[l].c\_str()));

ui->final\_2->addWidget(labelForTeamName);

fteamNameList[flength]=lteamNameList[l];

flength++;

}

l++;

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

*for* (int i=l; i<llength-1; i++) lteamNameList[i]=lteamNameList[i+1];

llength--;

labelForTeamName->setText(QString::number(l%2+1)+' '+QString::fromUtf8(lteamNameList[l+1].c\_str()));

*if* (llength>1) ui->lowerRound4->addWidget(labelForTeamName);

*else*

{

labelForTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(lteamNameList[l+1].c\_str()));

ui->final\_2->addWidget(labelForTeamName);

fteamNameList[flength]=lteamNameList[l+1];

flength++;

}

l++;

ui->resultLine->clear();

}

*if* (l==llength)

{

round++;

upperBracket=*true*;

l=0;

}

*break*;

*case* 4:

*if* (ui->resultLine->text()==QString::fromUtf8("1"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

*for* (int i=l+1; i<llength-1; i++) lteamNameList[i]=lteamNameList[i+1];

llength--;

labelForTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(lteamNameList[l].c\_str()));

ui->final\_2->addWidget(labelForTeamName);

fteamNameList[flength]=lteamNameList[l];

flength++;

l++;

ui->resultLine->clear();

}

*if* (ui->resultLine->text()==QString::fromUtf8("2"))

{

QDynamicLabel \*labelForTeamName = *new* QDynamicLabel(*this*);

labelForTeamName->setMaximumHeight(30);

QFont bold("Franklin Gothic Demi", 10);

labelForTeamName->setFont(bold);

*for* (int i=l; i<llength-1; i++) lteamNameList[i]=lteamNameList[i+1];

llength--;

labelForTeamName->setText(QString::number(flength+1)+' '+QString::fromUtf8(lteamNameList[l+1].c\_str()));

ui->final\_2->addWidget(labelForTeamName);

fteamNameList[flength]=lteamNameList[l+1];

flength++;

l++;

ui->resultLine->clear();

}

*if* (l==llength)

{

round++;

upperBracket=*true*;

l=0;

}

*break*;

}

}

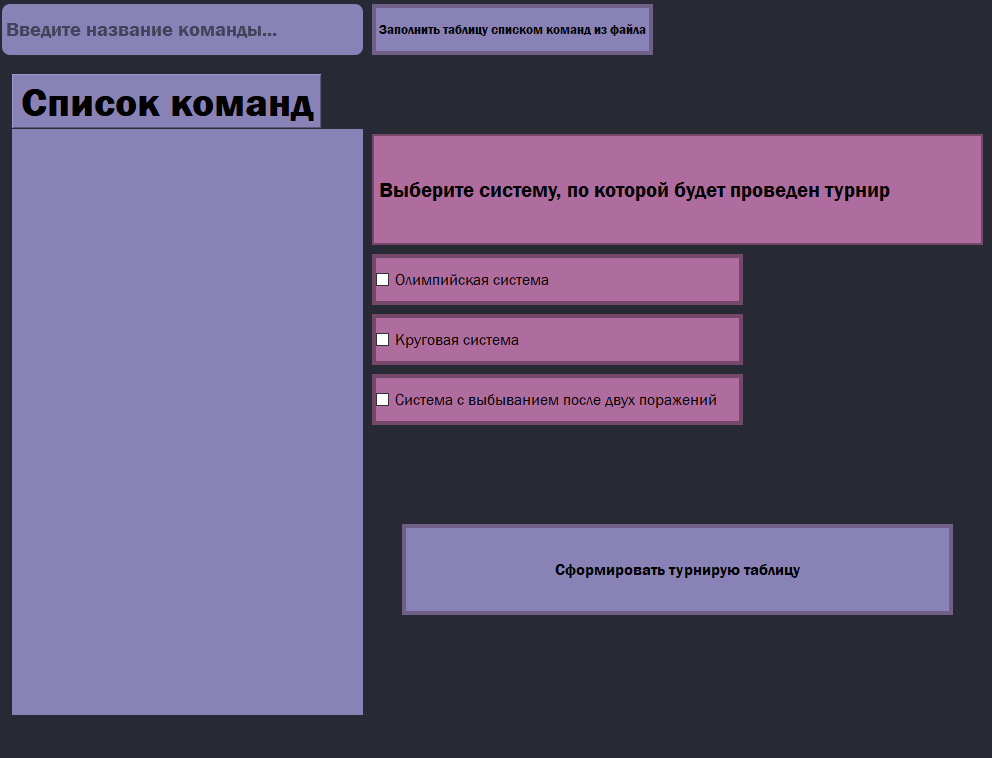
}

ui->resultLine->clear();

}

**3. Создание пользовательского дизайна**

Для того, чтобы подчеркнуть индивидуальность работы, был реализован полностью авторский дизайн, в стиле материал, который был применен ко всем окнам интерфейса.



Дизайн был реализован с помощью таблицы стилей StyleSheet.

QWidget

{

background-color:rgb(39, 41, 52);

}

QLabel

{

background-color:rgb(175, 108, 158);

border: 2px solid rgb(116, 72, 105);

}

QTableWidget

{

color: rgb(0, 0, 0);

background-color:rgb(137, 130, 182);

border: 0px solid #706087;

}

QTableWidget::item

{

background-color:rgb(137, 130, 182);

}

QHeaderView::section

{

background-color:rgb(137, 130, 182);

}

QTableWidget QTableCornerButton::section {background-color: rgb(137, 130, 182); }

QLineEdit

{

color: rgb(0, 0, 0);

background-color:rgb(137, 130, 182);

border: 3px solid #706087;

border-radius: 8px;

padding: 0 2px;

selection-background-color:rgb(160, 152, 205);

font-size: 20px;

}

QLineEdit:!focus

{

border: none;

}

QPushButton

{

color: rgb(0, 0, 0);

background-color:rgb(137, 130, 182);

border: 4px solid #706087;

padding: 0 2px;

}

QPushButton:pressed

{

color: rgb(0, 0, 0);

background-color:#706087;

border: 3px solid #706087;

padding: 0 2px;

}

QCheckBox

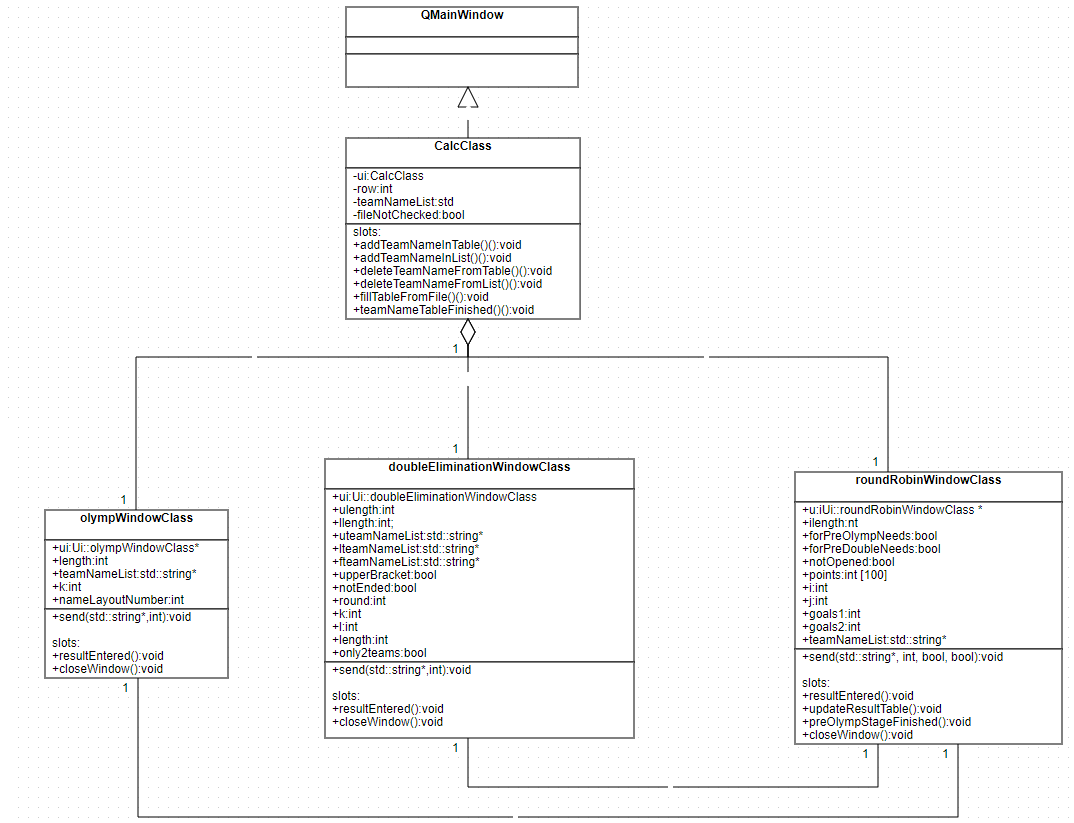
{

background-color:rgb(175, 108, 158);

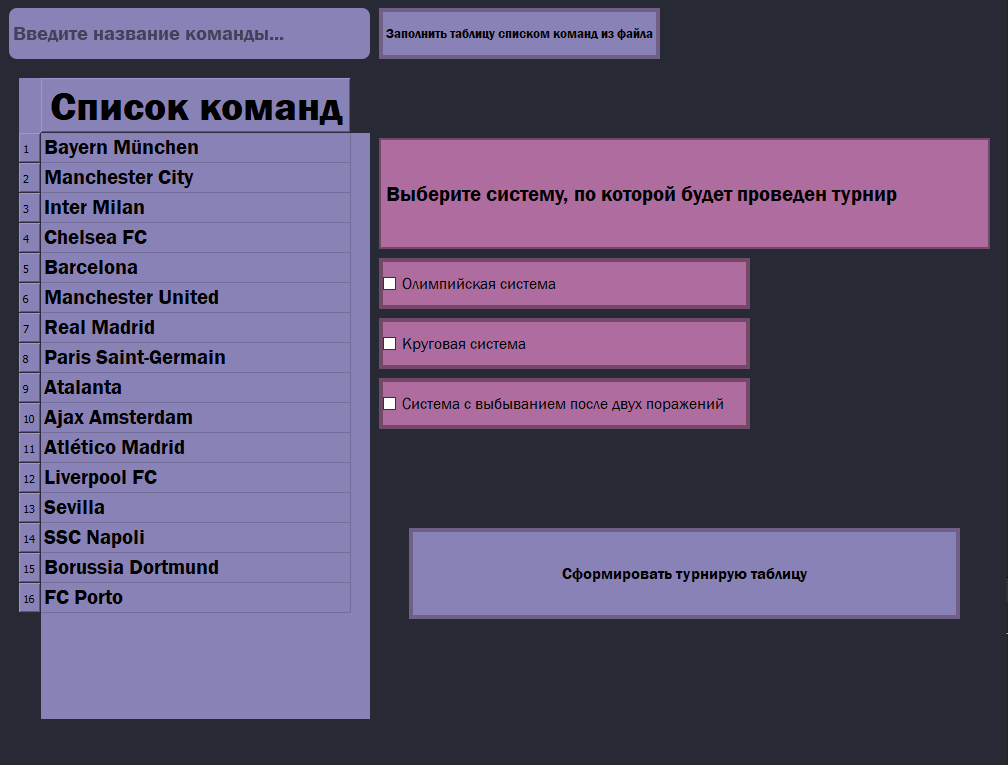
border: 4px solid rgb(116, 72, 105);

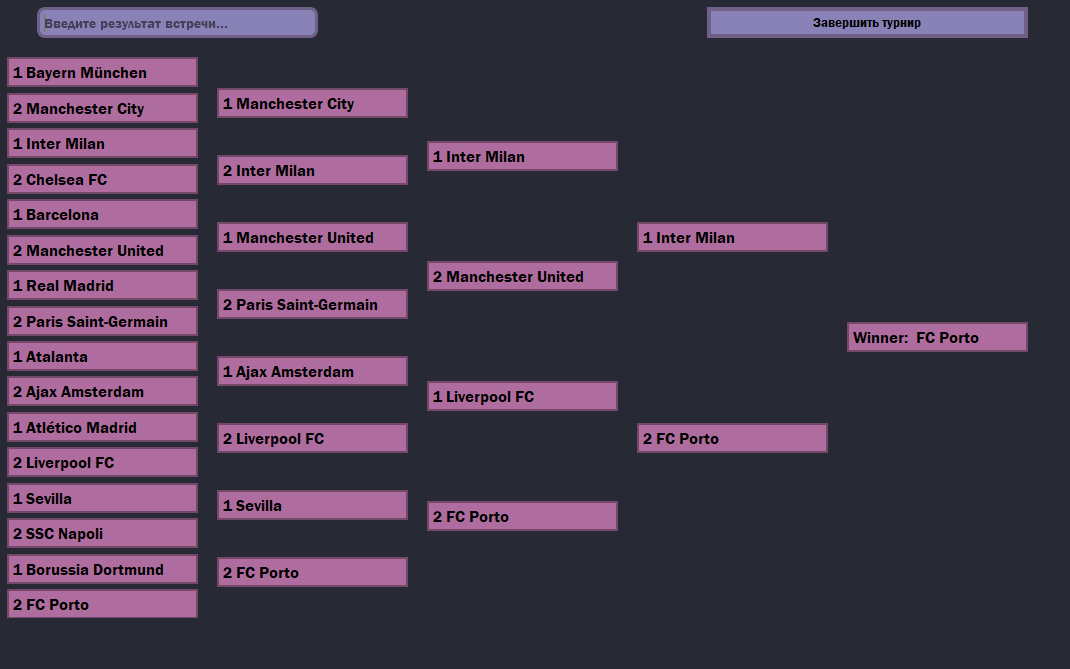
}

**Диаграмма классов**

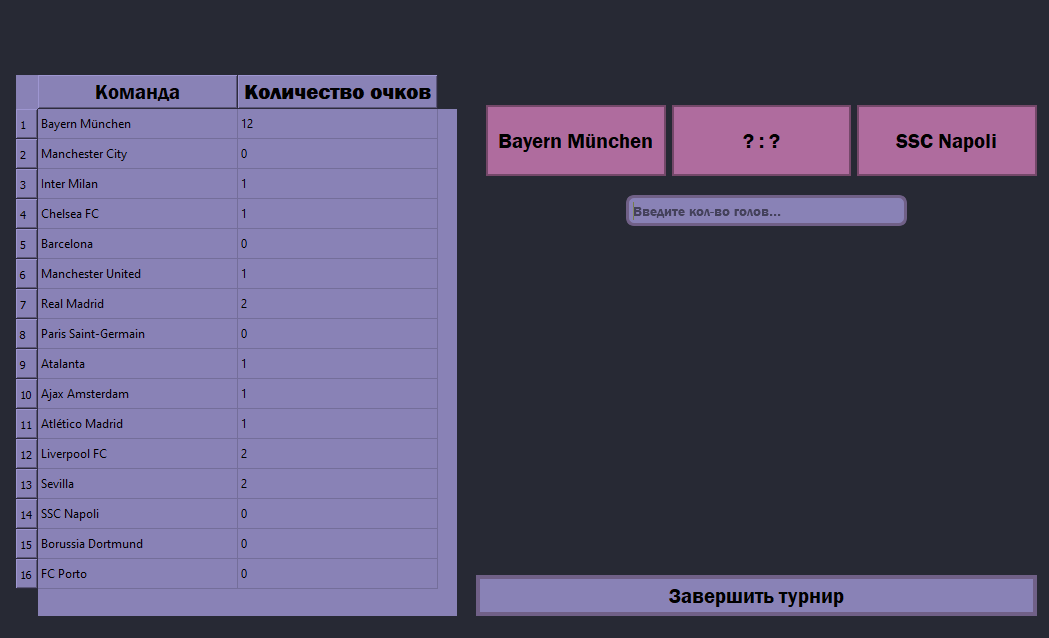
****

**Скриншоты работы программы**

****

****

****

****

## 2я часть – Задача Комивояжера

**Постановка задачи**

Реализовать программу решающую задачу Комивояжера: Требуется найти самый выгодный маршрут, проходящий через все города графа с последующим возвратом в исходный город. Посещать города повторно нельзя.

**Анализ задачи**

Для решения задачи Комивояжера будет использован метод перебора, так как он эффективен и прост в реализации.

Язык реализации программы - C++, как один из наиболее актуальных и многофункциональных на данный момент. Для реализации пользовательского интерфейса было принято решение использовать фреймворк QT, так как он предоставляет удобные инструменты для работы с интерфейсом. Для создания и тестирования программы использовалась интегрированная среда разработки QT Creator.

**Ход работы**

**1. Преобразование входных данных в форму, удобную для восприятия программой.**

Для хранения информации о дорогах и их длинах используется двумерный массив, первый индекс которого отвечает за номер города отправления, а второй- за номер города прибытия.

length[1][2]=28;

length[1][3]=13;

length[1][4]=15;

length[1][6]=18;

length[2][1]=28;

length[2][4]=20;

length[2][5]=21;

length[3][1]=13;

length[3][5]=30;

length[4][1]=15;

length[4][2]=20;

length[4][5]=39;

length[4][6]=31;

length[5][2]=21;

length[5][3]=30;

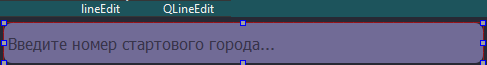
length[5][4]=39;

length[6][1]=18;

length[6][4]=31;

**2. Реализация решения задачи методом перебора.**

**2.1.** Для ввода номера начального города использовался объект интерфейса QLineEdit, которой представляет собой строку ввода данных.



**2.2.** Для решения задачи Комивояжера методом перебора была реализована функция, использующая рекурсию для более эффективного нахождения решения.

void find\_path(int n,int path,int kol)

{

checked[n]=true;

kol++;

way[kol]=n;

length[0][1]=0;

path+=length[way[kol-1]][n];

for (int i=1;i<=6;i++)

{

if ((length[n][i]!=0) and(checked[i]==false))

{

find\_path(i,path,kol);

}

}

int x=0;

for (int i=1;i<7;i++)

{

if (checked[i]==true) x++;

}

if (x==6 && length[n][startTown]!=0)

{

path+=length[n][startTown];

kol++;

way[kol]=startTown;

if (path<=min\_path)

{

min\_path=path;

min\_kol=kol;

int i=1;

while (way[i]!=0)

{

min\_way[i]=way[i];

i++;

}

}

}

checked[n]=false;

way[kol]=0;

kol--;

}

**2.3.** Для обработки и запуска функции поиска решения была реализована функция, срабатывающая при окончании ввода номера города.

void MainWindow::start()

{

if ((ui->lineEdit->text()).toInt()>=1 && (ui->lineEdit->text()).toInt()<=6)

{

for (int i=1;i<7;i++)

{

checked[i]=false;

}

for (int i=0;i<10;i++)

{

min\_way[i]=0;

way[i]=0;

}

ui->label\_path->setText("Длина маршрута:");

ui->label\_way->setText("Маршрут:");

startTown=(ui->lineEdit->text()).toInt();

checked[startTown]=true;

find\_path(startTown,0,0);

int i=1;

ui->label\_path->setText(ui->label\_path->text()+" "+QString::number(min\_path));

while (min\_way[i]!=0)

{

ui->label\_way->setText(ui->label\_way->text()+" "+QString::number(min\_way[i]));

i++;

}

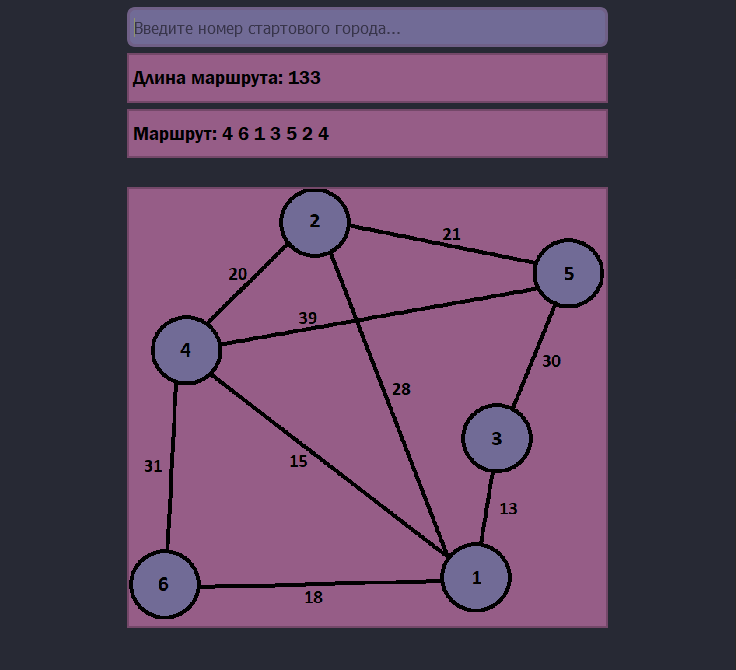
}

ui->lineEdit**->clear();**

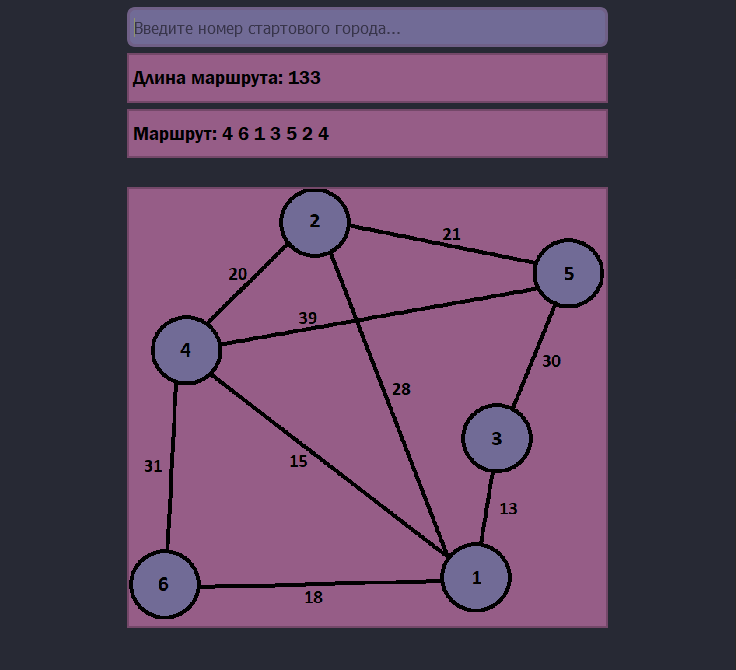
**}**

**3. Создание пользовательского дизайна**

Для того, чтобы подчеркнуть индивидуальность работы, был реализован полностью авторский дизайн, в стиле материал, который был применен к окну пользовательского интерфейса.

****

**Скриншоты результатов работы программы**

****

**GitHub**

[**https://github.com/sugarrrfqs/TvRab**](https://github.com/sugarrrfqs/TvRab)