C++Qt_day3 hw_1 천지인 키보드 만들기 과제 보고서

로빛 20기 인턴 현창석

목차

1. 과제 코드 설명 mainwindow.h 설명

hw_1/include/mainwindow.h* <Select Symbol>

arning: This file is not part of any project. The code model might have issues parsing this file properly.

arning: This file is outside the project directory.

```
1   #ifndef MYWIDGET_H
2   #define MYWIDGET_H
3
4   #include <QMainWindow>
5   #include <QTimer>
6   #include "ui_mainwindow.h"
7   class MyWidget : public QMainWindow
8   {
9       Q_OBJECT
10  public:
11      explicit MyWidget(QWidget *parent = nullptr);
12
13  protected:
14
15      void keyPressEvent(QKeyEvent *event) override;
16
17  public slots:
18      void saveFile();
19
20  private slots:
21      void on_pushButton_clicked();
22      void on_pushButton_2_clicked();
23      void on_pushButton_3_clicked();
24      void on_pushButton_4_clicked();
25      void on_pushButton_5_clicked();
26      void on_pushButton_6_clicked();
27      void on_pushButton_7_clicked();
28      void on_pushButton_8_clicked();
29      void on_pushButton_9_clicked();
30      void on_pushButton_10_clicked();
31      void on_pushButton_11_clicked();
32      void on_pushButton_12_clicked();
33      void on_pushButton_13_clicked();
34      void on_pushButton_14_clicked();
35      void on_pushButton_15_clicked();
36      void on_pushButton_16_clicked();
37      void on_pushButton_17_clicked();
38      void confirmPrevious();
39      void flashBackground();
40
41
42  private:
43      Ui::MainWindow *ui;    // ui 멤버 변수
44      bool ShiftOn = false;
45      int count = -1;
46      int count1 = 1;
47
48      QTimer *timer;  // QTimer 멤버 변수 추가
49
50      QString confirmedText;    // 이미 확정된 글자
51      QString tempText;         // 현재 버튼에서 선택 중인 임시 글자
52      int lastButtonClicked = -1; // 멤버 변수로 추가
53      QPalette originalPalette;
54
55
56
57  };
58
59  #endif
```

위 사진은 mainwindow.h의 코드를 캡쳐한 사진이다.
class mywidget을 Qmainwindow를 상속받아서 메인 윈도우 형태의 위젯을 출력하기 위해 선언하였다.

헤더파일이 중복 포함되는 것을 막기 위해 MYWIDGET_H을 선언하였다. 버튼과 상호작용하여 천지인 키보드를 구현하기 위해 사용한 함수를 이용하기 위해 선언하였다. 생성자를 설정하고 또한 엔터 키를 눌렀을 때 필요한 기능이 실행되도록 KEYPRESS 오버라이드를 호출해주었다. public slot에서는 키보드로 입력한 문장을 텍스트 파일에 저장하는 기능을 구현하는 함수를 선언하였다.
그 아래 선언된 17개의 함수는 각 버튼들을 클릭시 글자를 출력하거나 특정 기능을 수행하게 한다. confirmprevious 함수는 글자 버튼을 클릭하여 글자를 출력하다가 다른 글자 버튼을 누르면 이전에 출력되던 글자를 고정하고 다음 커서로 이동하는 기능이 있는 함수이다. 마지막으로 flashbackground는 글자가 입력된 후 고정되면 글자가 잘 입력되었다는 것을 나타내기 위해 입력창의 색이 잠깐 변하는 기능의 함수이다.
멤버 변수를 살펴보면 대문자 소문자를 shift가 활성화 된 상태에서 판별하기 위해 bool형 shifon을 선언하였다. 버튼을 클릭할 때마다 순서에 맞게 입력되는 글자를 바꿀 수 있게 하기 위한 count 변수를 선언하였다. 또한 글자의 다음 커서로 넘어가는 기능을 가지는 글자를 확정 시 사용할 함수를 각각 선언해주었다.

mainwindow.cpp 설명

```cpp
#include "../include/mainwindow.h"
#include <QtGlobal>
#include "ui_mainwindow.h"
#include <QPushButton>
#include <QVBoxLayout>
#include <QTimer>
#include <QFile>
#include <QTextStream>
#include <iostream>
#include "ui_mainwindow.h"
#include <QKeyEvent>


MyWidget::MyWidget(QWidget *parent)
    : QMainWindow(parent),
    ui(new Ui::MainWindow), count(0)
{
    ui->setupUi(this);  // 초기화

}

void MyWidget::flashBackground()
{

    originalPalette = ui->textEdit->palette();

    QPalette p = ui->textEdit->palette();
    p.setColor(QPalette::Base, QColor(Qt::yellow)); // 잠깐 노란색
    ui->textEdit->setPalette(p);


    QTimer::singleShot(200, this, [=]() {
        ui->textEdit->setPalette(originalPalette);
    });
}

void MyWidget::saveFile()
{
    QFile file("input_text.txt");

    if (file.open(QIODevice::WriteOnly | QIODevice::Text))
    {
        QTextStream out(&file);

        out << ui->textEdit->toPlainText() << "\n";

        file.close();
        std::cout << "input_text.txt 저장 완료\n";
    }
    else
    {
        std::cout << "오류 발생";
    }
}

void MyWidget::confirmPrevious()
{
    if (!tempText.isEmpty()) {
        confirmedText += tempText;
        tempText.clear();            // 임시 글자 초기화
    }
```

위 사진은 mainwindow.cpp의 코드를 캡쳐한 사진이다.

main.cpp 파일에는 화면을 출력하는 기능만 있기 때문에 설명을 생략하겠다.

QtCreator가 기본으로 생성하는코드는전부포인터(Ui::MainWindow *ui) 형태라서 ->를 주로 사용하여 프로그램을 작성하였다

MyWidget::MyWidget() 생성자 그리고 ui->setupUi(this);로 UI및 프로
그램의 초기 작업을 해준다. flashbackground 함수에서는 현재 textedit의 색상을 originalpalette에 다시 원래 색으로 되돌리기 위해 저장한다. 그 후 현재 palette를 p 변수에 복사한다. 그리고 setcolor 하여 배경 색을 노란색으로 변경한다.
QTimer::singleShot(200, this, [=]()은 200ms 후 한 번만 실행되는 타이머 설정이다. 따라서 200ms 후 palette를 색상이 변경되기 전으로 되돌린다.
savefile은 사용자가 입력한 문장을 txt 파일에 저장하는 기능을 하는 함수이다. 먼저 input_text.txt 라는 파일 객체를 생성하고 파일 쓰기 모드르 파일을 오픈한다. 그 후 사용자가 입력한 문장이 출력되어 있는 textEdit안의 전체 텍스트를 파일에 쓰고 줄바꿈해준다. 파일 쓰기가 완료되었다면 터미널에 저장완료 메시지가 출력된다. 파일 열기가 실패했을 경우에는 오류가 발생했다는 메시지를 터미널에 출력한다.
confirmprevious 함수는 임시로 출력한 텍스트를 확정짓는 역할을 한다. 임시 텍스트가 저장된 temptext가 비어있지 않으면 확정되어 있는 confirmedtext와 temptext를 합쳐 임시 텍스트와 확정된 텍스트를 통합한다. 그 후 temptext를 초기화한다.

```
64
65
66  void MyWidget::on_pushButton_clicked()
67  {
68      if (lastButtonClicked != 1) {confirmPrevious(); count =-1; flashBackground();}
69      lastButtonClicked = 1;
70
71      count++;
72      int choice = count % 4;
73      switch(choice){
74      case 0: tempText = "."; break;
75      case 1: tempText = ","; break;
76      case 2: tempText = "?"; break;
77      case 3: tempText = "!"; break;
78      }
79
80      if (ShiftOn) tempText = tempText.toUpper();
81      ui->textEdit->setPlainText(confirmedText + tempText);
82      ui->textEdit->moveCursor(QTextCursor::End);
83  }
84
85
86  void MyWidget::on_pushButton_2_clicked()
87  {
88      if (lastButtonClicked != 2) {confirmPrevious(); count =-1;flashBackground();}
89      lastButtonClicked = 2;
90
91      count++;
92      int choice = count % 3;
93      switch(choice){
94      case 0: tempText = "a"; break;
95      case 1: tempText = "b"; break;
96      case 2: tempText = "c"; break;
97      }
98
99      if (ShiftOn) tempText = tempText.toUpper();
100     ui->textEdit->setPlainText(confirmedText + tempText);
101     ui->textEdit->moveCursor(QTextCursor::End);
102 }
103
```

이 사진은 글자를 입력하는 함수를 캡쳐한 사진이다.
버튼이 클릭된 상태라면 confirmprevious 함수를 실행해서 임시 텍스트와 확정 텍스트를 합쳐주고 flashbackground 함수로 입력 창의 색을 잠시 바꾼다. 버튼을 클릭할 때마다 count가 1씩 더해지며 이에 따라 출력되는 임시텍스트가 바뀐다. shift 버튼이 입력된 상태라면 임시텍스트를 대문자로 변경한다.
특정 기능을 제외한 문자를 입력하는 버튼들은 이와 같은 알고리즘을 가진다.

```
// 4번 버튼 - 지우기
void MyWidget::on_pushButton_4_clicked()
{
    if (!tempText.isEmpty()) {
        // 아직 확정되지 않은 글자가 있으면 그냥 삭제
        tempText.clear();
    } else if (!confirmedText.isEmpty()) {
        // 마지막 확정 글자 삭제
        confirmedText.chop(1);
    }

    ui->textEdit->setPlainText(confirmedText + tempText);
    ui->textEdit->moveCursor(QTextCursor::End);
}
```

이 4번 버튼이 클릭되면 위와 같은 함수가 실행된다. 임시 텍스트가 있다면 지우고 확정된 글자라면 문장의 가장 뒤에 있는 마지막 글자를 삭제한다.

```
// 8번 버튼 - 줄바꿈
void MyWidget::on_pushButton_8_clicked()
{
    confirmPrevious();

    confirmedText += "\n";
    tempText.clear();

    ui->textEdit->setPlainText(confirmedText + tempText);

    QTextCursor cursor = ui->textEdit->textCursor();
    cursor.movePosition(QTextCursor::End);
    ui->textEdit->setTextCursor(cursor);
    ui->textEdit->ensureCursorVisible();

    lastButtonClicked = 8;
}
```

8번 버튼을 클릭하면 사용자가 입력하고 있는 문장에서 줄바꿈을 실행한다. 줄바꿈 명령을 텍스트에 추가하고 바꾼 줄의 처음으로 커서의 위치를 옮겨주었다.

```
277
278
279 void MyWidget::on_pushButton_12_clicked()
280 {
281     ShiftOn = !ShiftOn;  // 클릭할 때마다 상태 토글
282     QList<QPushButton*> buttons = {
283         ui->pushButton, ui->pushButton_2, ui->pushButton_3,
284         ui->pushButton_5, ui->pushButton_6, ui->pushButton_7,
285         ui->pushButton_9, ui->pushButton_10, ui->pushButton_11
286     };
287
288     if(ShiftOn) {
289
290         for(auto btn : buttons) {
291             QString text = btn->text();
292             btn->setText(text.toUpper());
293         }
294     } else {
295
296         for(auto btn : buttons) {
297             QString text = btn->text();
298             btn->setText(text.toLower());
299         }
300     }
301 }
```

위 사진은 천지인 키보드에서 shift 역할을 하는 버튼을 클릭했을 때 작동하는 함수를 캡쳐한 사진이다. 버튼이 클릭될 때마다 상태가 토글되고 참이라면 버튼의 텍스트를 대문자로 바꾸고 그렇지 않다면 소문자로 유지한다.

```
335
336
337 void MyWidget::keyPressEvent(QKeyEvent *event)
338 {
339     if (event->key() == Qt::Key_Return || event->key() == Qt::Key_Enter) {
340
341         saveFile();
342
343         event->accept();
344         return;
345     }
346
347     // 엔터가 아닌 키는 무시
348     event->ignore();
349 }
350
351
352
```

위 함수는 사용자가 엔터키를 누르면 입력한 문장을 텍스트 파일을 저장하는 함수를 실행시키는 역할을 한다. 엔터키 밑 숫자 키패드 쪽의 엔터키를 누르면 savefile 함수를 실행시켜 텍스트 파일에 입력한 문장을 저장한다.
엔터키가 아닌 키는 ignore로 무시해준다.