HW1 보고서

로빛 20기 예비인턴

2025407006 로봇학부 모시온

목차

1. 개요

본 프로젝트는 휴대폰과 같은 모바일 장치에서 영어 또는 한글 문자를 입력하기 위해 구성된 천자인 배열 구성의 키보드를 QT와 C++로 직접 재현하여 이를 구현하는 프로젝트

%%영어와 달리 자음과 모음의 결합이 존재하는 한글 특성상
아래아자와 음절의 결합에 따른 형태 변화의 어려움으로 영어 및 기본 기호 입력 재현만을 구현하였습니다.%%

구체적인 설명 이전에 각각의 키 배열을 설명합니다.

B1: .,?!

B2: abc

B3: def

B4: ghi

...

와 같이 B1 ~ B10까지 구성된 기본 입력 버튼(B10은 0 입력)
과 이외의 ,(콤마), Shift, Enter, Backspace를 이름과 동일하게 코드를 구성하였습니다.

2. .h/.hpp 헤더파일 설계

```
#include <QMainWindow>
#include <QTimer>
#include <QVector>
#include <QString>
```

각각의 모듈에 대해 설명합니다.

1. QMainWindow : QT 소프트웨어와 창(윈도우)의 상속 모듈
2. QTimer: 하나의 버튼에 3개 이상의 문자 할당된 천자인 배열은
   각 버튼의 입력 지연시간을 통해 원하는 문자 할당
   (예: abc, 버튼 빠르게 2번 클릭 -> b, 1초 뒤 클릭 -> a)
3. QVector: 한/영의 변환에 따라 버튼 하나에 할당되는 문자의 동적 할당을 하려하였으나 미구현, 문자 저장 및 입력란 담당
4. QString: 영어 입력 버튼의 Shift 처리와 문자열 입력 담당

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
```

MainWindow의 Class에 대하여 QT의 제공되는 시그널 및 슬롯을 활성화하기 위하여 Q_OBJECT를 사용

Class에 대한 생성자 및 소멸자를 적용하기 위해 public 작성

```
private slots:
    void on_B1_clicked();
    void on_B2_clicked();
    void on_B3_clicked();
    void on_B4_clicked();
    void on_B5_clicked();
    void on_B6_clicked();
    void on_B7_clicked();
    void on_B8_clicked();
    void on_B9_clicked();
    void on_B10_clicked();

    void on_Shift_clicked();
    void on_Space_clicked();
    void on_Comma_clicked();
    void on_Enter_clicked();
    void on_Backspace_clicked();

    //천지인 키보드 문자 확정용
    void commit();
```

UI의 생성된 버튼 동작 함수

+ 시간 지연 감지를 통한 문자 확정 함수

(commit)

```
private:
    Ui::MainWindow *ui;

    QVector<QString> button_assign;
    QTimer *commit_Time;
    //1초 기준

    //문자열 제작
    int button_final;
    //버튼 그룹 내 인덱스
    int button_index;
    bool shift;

    QString commit_Text;


    void generate_char(int buttonIndex);
    void display_string();
    void shifting();
    void particular_symbols(const QChar &c); //Space/Comma/Enter
```

UI 위젯 참조를 위한 선언과 문자의 동적 할당 및 문자열(".,?!", "abc",...)을 저장하기 위하여 호출한 모듈을 사용해 button_assign 선언

QTimer를 통해 지정 지연 시간 이후 문자 확정(1초)

가장 마지막으로 입력된 문자 번호를 저장하는 button_final

해당 문자 번호를 통한 사전에 할당된 버튼 UI 위치 변수 button_index

Shift 감지를 위한 불리언 변수 Shift

또한, 확정된 문자열을 처리하기 위한 commit_Text

문자 출력을 위한 함수

generate_char: 문자 위치를 통한 문자 출력

display_string: 입력되어가는 문자열(Enter 안 누른 상태) 출력 담당

shifting: Shift 이후 처리 담당

콤마 및 backspace와 enter 처리를 위한 particular_symbols

3. .c/.cpp 소스파일 설계

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QFile>
#include <QTextStream>
#include <QTextCursor>
```

기본 윈도우 사용과 UI 사용을 위한 모듈 호출

+

1. txt파일 저장을 위한 모듈 QFile
2. 저장할 문자열을 한줄씩 처리하기 위한 모듈 QTextStream
3. 공백 또는 기타 입력에 따른 입력 칸 이동을 위한 QTextCursor

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    commit_Time = new QTimer(this);
    button_final = -1;
    button_index = 0;
    shift = false;

    button_assign.resize(10);
    button_assign[0] = ".,?!";
    button_assign[1] = "abc";
    button_assign[2] = "def";
    button_assign[3] = "ghi";
    button_assign[4] = "jkl";
    button_assign[5] = "mno";
    button_assign[6] = "pqrs";
    button_assign[7] = "tuv";
    button_assign[8] = "wxyz";
    button_assign[9] = "0";
```

UI 위젯을 초기화 및

부모 위젯을 설정하며 각각의 위젯을 모두 초기화합니다.

그리고 사전에 Vector의 크기를 10으로 할당하며 이에 따라 각각의 버튼 10개에 해당되는 문자열을 저장합니다.

```
commit_Time->setSingleShot(true);
//1초 기준 내 문자변경 간격
commit_Time->setInterval(1000);
connect(commit_Time,SIGNAL(timeout()),this,SLOT(commit()));

commit_Text = "";
shifting();
display_string();
```

함수 commit_Time에서는 버튼 한 번 누룰 때마다 commit_Time 함수를 한 번 동작, 1초를 간격으로 commit함수를 호출하여 문자를 결정.

또한, 초기 코드 실행시 비어있는 칸을 위해 버튼의 라벨 초기화와 텍스트를 비워둡니다.

```
void MainWindow::generate_char(int idx)
{
    if(button_final!=-1 && button_final!=idx) commit();

    if(button_final==idx){
        QString t=button_assign[idx];
        if(t.length()>0) button_index=(button_index+1)%t.length();
        else button_index=0;
    }else{
        button_final=idx;
        button_index=0;
    }

    display_string();
    commit_Time->start();
}
```

전달받은 문자의 UI 위치에 따라 지정된 문자 이외의 다른 문자 버튼을 누르는 경우 마지막 문자를 확정하여 고정시키도록합니다.
하지만 같은 버튼 시간내 클릭시 계속 주소를 1씩 증가시키며 버튼 하나에 할당된 3개 이상의 문자를 순환할 수 있도록 구성합니다.
또한, 지금 입력을 고려하는 문자를 display_string을 통해 출력합니다.

```
void MainWindow::display_string()
{
    QString s=commit_Text;
    if(button_final!=-1){
        QString t=button_assign[button_final];
        if(t.length()>0){
            QChar c = t[button_index];
            if(shift && c.isLetter()) c=c.toUpper();
            s+=c;
        }
    }

    ui->textEdit->blockSignals(true);
    ui->textEdit->setPlainText(s);
    QTextCursor cur=ui->textEdit->textCursor();
    cur.movePosition(QTextCursor::End);
    ui->textEdit->setTextCursor(cur);
    ui->textEdit->blockSignals(false);
}
```

commit_Text를 먼저 호출하여 확정 문자를 먼저 보이도록 하였으며, 가장 마지막 입력 버튼을 감지하는 button_final의 값을 활용해 미리볼 수 있도록 구성하였습니다.

(엔터 입력 X, 1초 이내의 시간 동안 현재 입력 시도했던 문자 시각화)

또한 Shift입력을 감지할 시 QString 모듈을 통해 버튼의 내부 속성 문자를 대문자 또는 소문자로 변경할 수 있도록 하였습니다.

(QString 사용과 관련하여 버튼의 내부 문자 속성 변환에 대해 직접 구현하는 방식으로 해결하는 것이 목표였으나 디버깅의 미숙으로 모듈 활용을 하였습니다.)

Textedit에 문자 입력 중인 동안에는 다른 입력의 실수로 인한 오류 방지를 위하여 신호 차단, 커서 초기화, 커서 옮기기, 신호 차단을 하였습니다.

```
void MainWindow::particular_symbols(const QChar &c)
{
    if(button_final!=-1) commit();
    if(!c.isNull()){
        commit_Text+=c;
        display_string();
    }
}
```

.,?!과 같은 특수문자 클릭에 따른 출력 코드입니다.

버튼의 입력이 끝남을 감지하고 이후 commit함수 호출을 통해

문자 확정을 합니다.

가장 중요한 것은 해당 함수는 Enter, Backspace 기능또한 담당하므로

우선 처리될 수 있도록 버튼 입력이 되면 무조건 실행합니다.

```
void MainWindow::commit()
{
    if(button_final==-1) return;

    QString t=button_assign[button_final];
    if(t.length()>0){
        QChar c=t[button_index];
        if(shift && c.isLetter()) c=c.toUpper();
        commit_Text+=c;
    }

    button_final=-1;
    button_index=0;
    if(commit_Time->isActive()) commit_Time->stop();

    display_string();
}
```

문자 확정 함수입니다.

버튼 반복 클릭 또는 1초의 시간 지남 이전에 반환을 바로 하여

문자확정이 이루지 않을 수 있도록 하며 아닌 경우에는 문자 확정을

위해 버튼 위치에 따른 문자 c를 통하여 문자를 받으며 commit_Text에

이를 더하여 누적하여 문자열을 이루도록 합니다.

문자 확정 이후에는 시간을 다시 초기화해야 다음 문자 대기가 가능하므로 stop으로 초기화하며 미리보기 함수를 가동합니다.

```
void MainWindow::shifting()
{
    ui->B1->setText(button_assign[0]);
    for(int i=1;i<=8;i++){
        QString t = button_assign[i];
        if(shift) t=t.toUpper();
        if(i==1) ui->B2->setText(t);
        else if(i==2) ui->B3->setText(t);
        else if(i==3) ui->B4->setText(t);
        else if(i==4) ui->B5->setText(t);
        else if(i==5) ui->B6->setText(t);
        else if(i==6) ui->B7->setText(t);
        else if(i==7) ui->B8->setText(t);
        else if(i==8) ui->B9->setText(t);
    }
    ui->B10->setText(button_assign[9]);
}
```

Shift 함수입니다.

for 반복문을 통해 해당 함수 가동 시 대문자가 할당할 수 있는 모든 버튼의 문자를 setText를 통해 대문자로 변경합니다.

(1부터 시작하여 특수문자 변경을 예방합니다.)

```
void MainWindow::on_B1_clicked(){generate_char(0);}
void MainWindow::on_B2_clicked(){generate_char(1);}
void MainWindow::on_B3_clicked(){generate_char(2);}
void MainWindow::on_B4_clicked(){generate_char(3);}
void MainWindow::on_B5_clicked(){generate_char(4);}
void MainWindow::on_B6_clicked(){generate_char(5);}
void MainWindow::on_B7_clicked(){generate_char(6);}
void MainWindow::on_B8_clicked(){generate_char(7);}
void MainWindow::on_B9_clicked(){generate_char(8);}
void MainWindow::on_B10_clicked(){generate_char(9);}

void MainWindow::on_Shift_clicked(){shift=!shift; shifting(); display_string();}
void MainWindow::on_Space_clicked(){particular_symbols(' ');}
void MainWindow::on_Comma_clicked(){particular_symbols(',');}
```

각 입력 버튼 가동 함수입니다.

```
void MainWindow::on_Enter_clicked(){
    if(button_final!=-1) commit();
    commit_Text+='₩n';
    display_string();
    ui->label->setText(commit_Text);

    //파일입출력
    QFile f("output.txt");
    if(f.open(QIODevice::WriteOnly|QIODevice::Append|QIODevice::Text)){
        QTextStream out(&f);
        out<<commit_Text<<"₩n";
        f.close();
    }
    commit_Text="";
}
//지우기 버튼
```

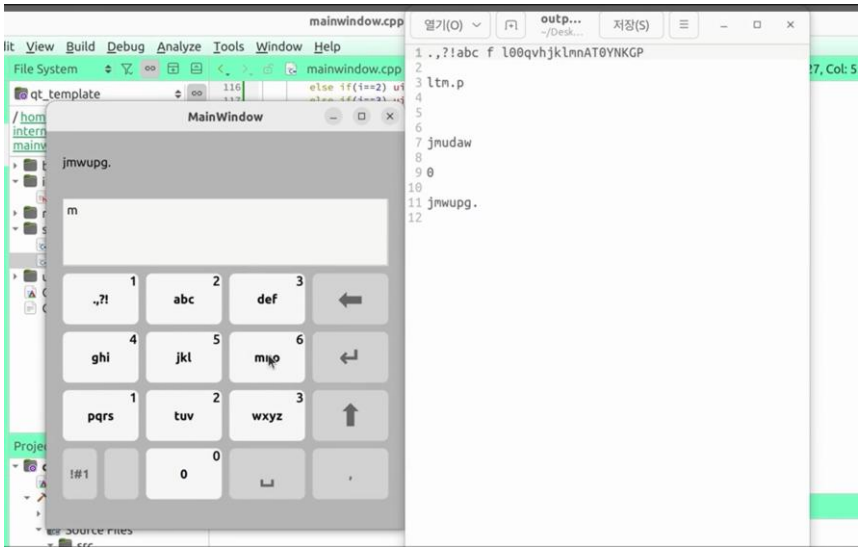
만일 Enter입력 시 현재 미리보기의 누적된 문자열을 txt로 저장할 수 있도록 문자확정을 마지막으로 실행 후 줄넘김 입력을 통해 파일의 마지막 줄에 누적 저장을 이루도록 합니다,

호출한 모듈 QFile을 이때 사용하여 파일 저장을 할 수 있도록 하며, 파일 저장을 위한 코드는 아래의 블로그를 참고하여 제작하였습니다.

https://blog.naver.com/browniz1004/220707555543

파일을 쓰기모드로 생성 또는 열며 누적된 문자열을 삽입하고 줄넘김을 최후에 하며 마칩니다.

4. 실행 결과



엔터 입력 후 문자 확정을 이루며 오른쪽의 파일에 저장됩니다.