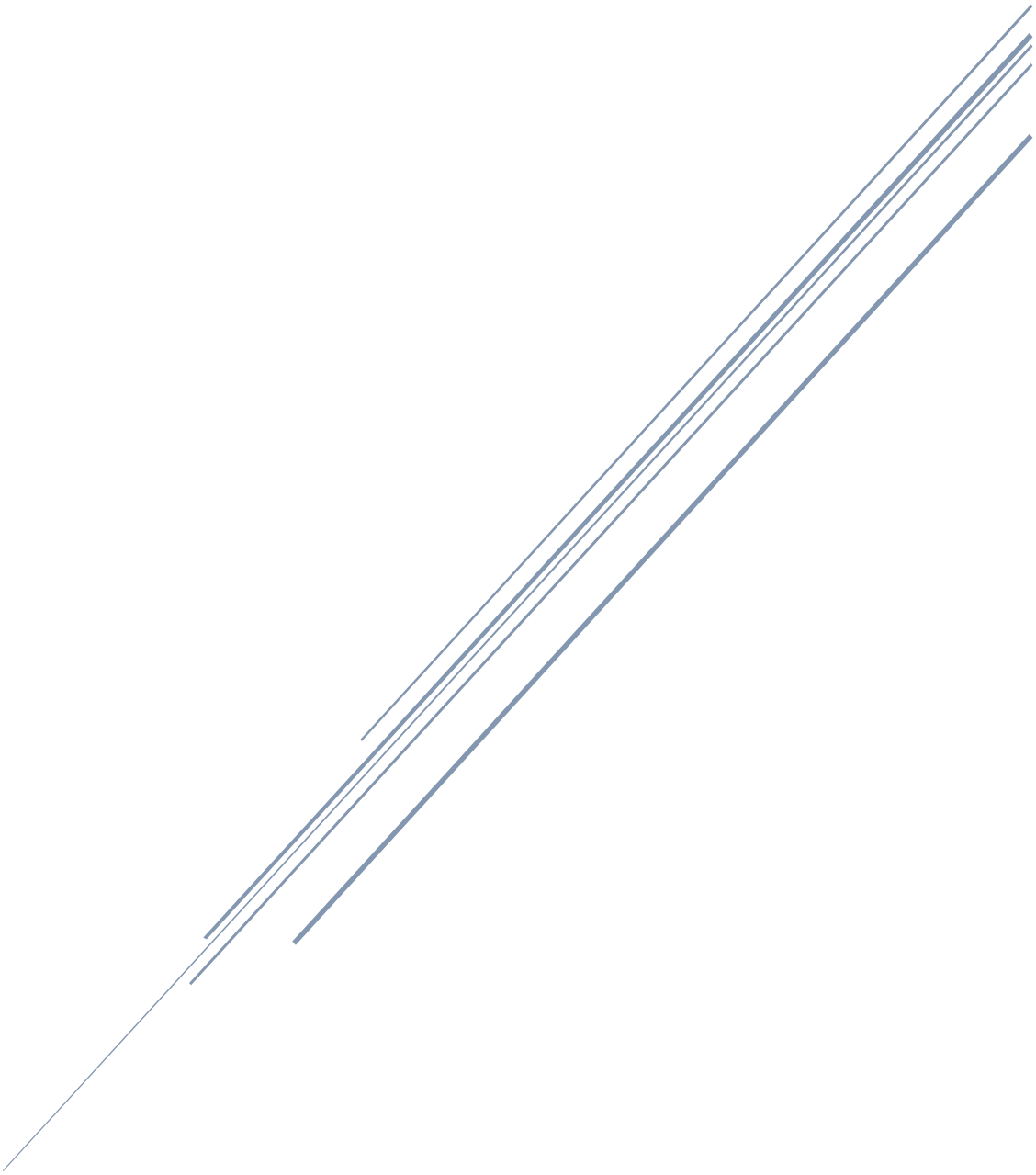
PROGRAMOWANIE OBIEKTOWE

Dokumentacja do projektu zaliczeniowego



Marcin Osiński, 235267
Piątek 11.15

Spis treści

1. Wstęp

Niniejszy program generuje losowo 4 zestawy, wektory, 10 liczb. Co 30 sekund następuje usunięcie pierwszej liczby, wygenerowanie nowej i dopisanie jej na końcu wektora. Generowanie nowej liczby działa według zasady, że każda następna może się różnić od poprzedniej o wartość z przedziału: <-0,5; 0,5>

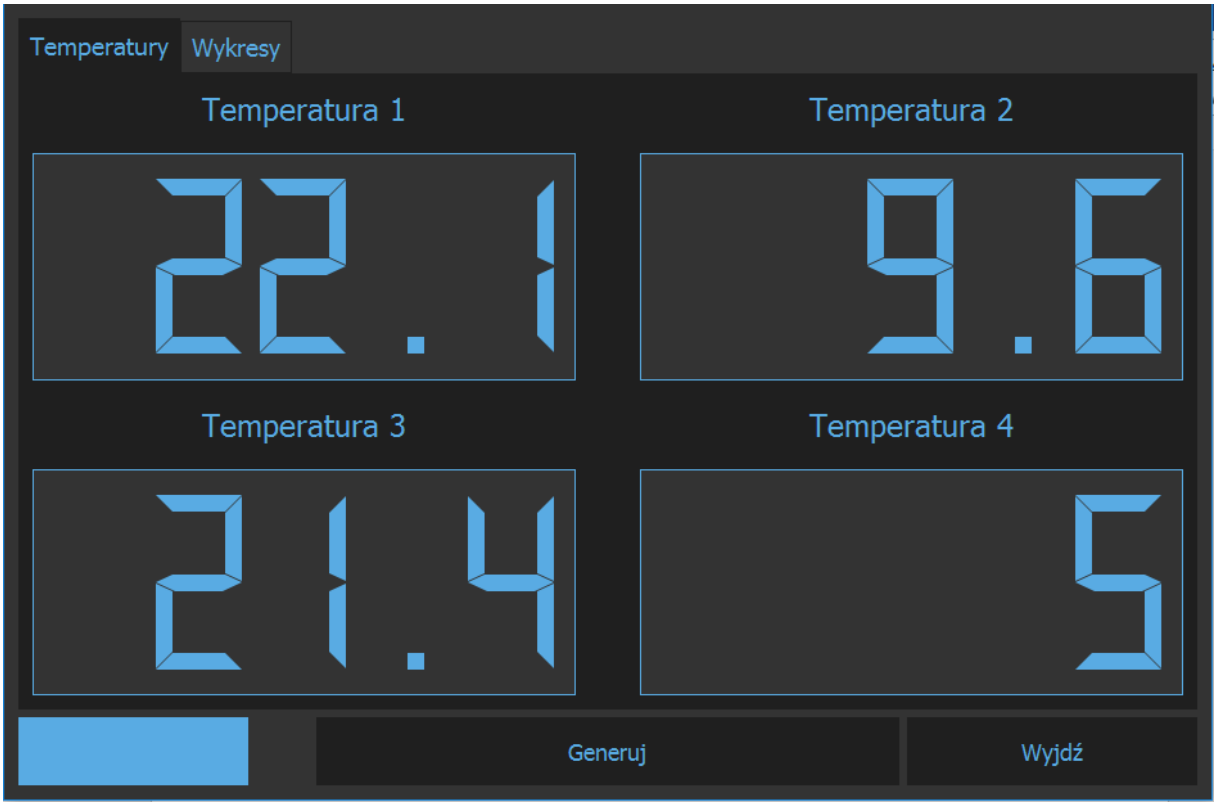
Generowane liczby są także zapisywane w pliku tekstowym.

Interfejs użytkownika można rozdzielić na dwie składowe: zakładki, które prezentują dane, oraz dolny pasek wspólny dla obu zakładek z paskiem postępu do wygenerowania nowej wartości po lewej (pokazuje stan zegara, który odlicza czas do wygenerowania nowej liczby), przyciskiem „Generuj" na środku, po przyciśnięciu którego następuje natychmiastowe wygenerowanie nowej wartości (pasek postępu i zegar zliczający zostają wtedy wyzerowane), oraz przycisku „Wyjdź", którego działanie jest oczywiste.

W pierwszej z zakładek („Temperatury") prezentowane są ostatnie wartości z każdego z wektorów.

W drugiej znajdują się wykresy prezentujące wartości całych wektorów na przestrzeni czasu.

Wygląd programu prezentuję na poniższych ilustracjach:



Rysunek 1: wygląd zakładki "Temperatury"

Rysunek 2: wygląd zakładki „Wykresy”

Aby uzyskać widoczny efekt posłużyłem się Qt. Ponadto, do wykonaniu wykresów użyłem biblioteki qCustomPlot. Generowaniem liczb zajmuje się klasa, którą sam napisałem.

Zatem łącznie w programie znajdują się trzy klasy:

1. SensorSimulator – odpowiada za generowanie liczb,
2. qCustomPlot – odpowiada za wykres,
3. MainWindow – klasa główna programu, tworzona automatycznie przez IDE Qt Creator, dziedzieczy publicznie po QMainWindow (jednej z klas składowych Qt)

W niniejszej dokumentacji nie będę omawiał klasy qCustomPlot, jako że tylko jej używałem (dokumentację do niej można znaleźć pod adresem qcustomplot.com).

2. Klasa SensorSimulator

Klasa SensorSimulator do działania potrzebuje dołączenia następujących bibliotek:

1. iostream
2. ctime
3. cector
4. cstream
5. clgorithm

```
class SensorSimulator
{
private:
    double temp1, temp2, temp3, temp4;

    vector<double>temps1;
    vector<double>temps2;
    vector<double>temps3;
    vector<double>temps4;

    fstream file;

    void write_to_file(vector<double> &vector_to_write);
    double delta();

public:
    SensorSimulator();

    void generate_temp(double previous_temp1, double previous_temp2,
                       double previous_temp3, double previous_temp4,
                       unsigned int how_many);

    double get_temp1(), get_temp2(), get_temp3(), get_temp4();
    double get_temps1(int index), get_temps2(int index),
           get_temps3(int index), get_temps4(int index);
    double get_min(), get_max();
};
```

Deklaracja klasy SensorSimulator

double temp…

Przechowuje ostatnią z wygenerowanych liczb

vector<double> temps…

Wektory przechowujące wygenerowane liczby

fstream file

Obiekt reprezentujący plik, do którego zapisywane są wektory

void write_to_file(vector<double> &vector_to_write)

Zapisuje wektor do pliku reprezentowanego przez obiekt fstream file

double delta()

Zwraca liczbę z przedziału <-0,5; 0,5>

SensorSimulator()

Konstruktor, losuje pierwsze wartości i zapisuje je w zmiennych temp1, temp2, itd.

void generate_temp(double previous_temp1, double previous_temp2, double previous_temp3, double previous_temp4, unsigned int how_many)

Generuje kolejne wartości na podstawie poprzednich, zapełniając wektory. Argument how_many ustala ile liczb ma się w nich mieścić

double get_temp…()

Zwraca wartość zmiennej temp…

double get_temps…(int index)

Zwraca wartość wektora z miejsca, na które wskazuje index

double get_min()

Zwraca najmniejszą wartość ze wszystkich wektorów

double get_max()

Zwraca największą wartość ze wszystkich wektorów

3. Klasa MainWindow

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

public slots:
    void newSecond();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

private:
    Ui::MainWindow *ui;

    SensorSimulator sensor;

    int time;

    QDateTime qtime;
    QTimer *timer;
    QScreen *screen;
    QRect screenGeometry;
    QString xAxisValue[10];

    void drawLinesOnGraph();
};
```

Deklaracja klasy MainWindow

Klasa MainWindow do działania potrzebuje dołączenia:

1. QMainWindow
2. QScreen
3. QRect
4. QTimer
5. QTime
6. Sensorsimulator.h
7. Qcustomplot.h

W deklaracji można zaobserwować sekcje public i private slots. Są to specjalne mechanizmy Qt do obsługi tak zwanych sygnałów, które są generowane np. przy wciśnięciu przycisku, czy innych zdarzeniach.

Makro Q_OBJECT na początku ciała klasy wskazuje, że dana klasa korzysta z mechanizmu sygnałów i slotów i musi być kompilowana z użyciem tzw. moc (Meta-Object Compiler). Za dokumentacją Qt:

> The Meta-Object Compiler, moc, is the program that handles Qt's C++ extensions.
>
> The moc tool reads a C++ header file. If it finds one or more class declarations that contain the Q_OBJECT macro, it produces a C++ source file containing the meta-object code for those classes. Among other things, meta-object code is required for the signals and slots mechanism, the run-time type information, and the dynamic property system.
>
> Dokumentacja Qt

explicit MainWindow(QWidget *parent = 0)

Konstruktor klasy MainWindow. *parent = 0 oznacza standardowego rodzica dla tworzonego okna. Słowo kluczowe explicit oznacza, że przy użyciu tego konstruktora nie będą przeprowadzane niejawne konwersje.

W konstruktorze tworzona jest część GUI programu (reszta jest generowana automatycznie przez QtDesigner). Inicjalizowany jest zegar odliczający czas do wygenerowania nowych liczb, a także generowane jest pierwsze dziesięć wartości, które następnie są wyświetlane.

~MainWindow()

Destruktor, niszczy ui.

void newSecond();

Slot obsługujący sygnał timeout od timera, który pojawia się co sekundę. Gdy naliczone zostanie 30 sekund timer oraz wskaźnik (progress bar) są zerowane i generowana są nowe wartości.

void on_pushButton_clicked()

Slot obsługujący wciśnięcie przycisku, w tym wypadku przycisku „Generuj”. Zeruje timer oraz wskaźnik (progress bar) i generuje nowe wartości.

void on_pushButton2_clicked()

Slot obsługujący wciśnięcie przycisku „Wyjdź”. Zamyka program

Ui::MainWindow *ui

Wskaźnik na interfejs użytkownika generowany przez QtDesigner. Dzięki niemu można w kodzie odwołać się do elementów stworzonych przy pomocy edytora WYSIWYG jakim jest QtDesigner.

SensorSimulator sensor

Obiekt klasy SensorSimulator omawianej powyżej. Odpowiada za generowanie liczb.

int time

Zmienna typu int licząca ile razy wystąpił sygnał timeout od timera.

QDateTime qtime

Zmienna przechowująca aktualną godzinę (typ QDateTime pozwala również na przechowywanie informacji o dacie)

QTimer *qtimer

Wskaźnik na obiekt typu QTimer, który odlicza czas, generując co sekundę sygnał timeout.

QScreen *screen

Wskaźnik na obiekt typu QScreen przechowujący informację o ekranie.

QRect screenGeometry

Obiekt typu QRect przechowuje informacje o rozmiarach ekranu. Razem z obiektem typu QScreen pozwala ustalić aktualny rozmiar ekranu, co jest wykorzystywane przy skalowaniu wielkości i rozmieszczenia paska postępu i obu przycisków.

void drawLinesOnGraph()

Kreśli linie na wykresie. Dba o odpowiednie wyskalowanie osi.

4. Bibliografia

1. Jerzy Grębosz „Symfonia C++ Standard”
2. Dokumentacja Qt doc.qt.io
3. Forum stackoverflow.com