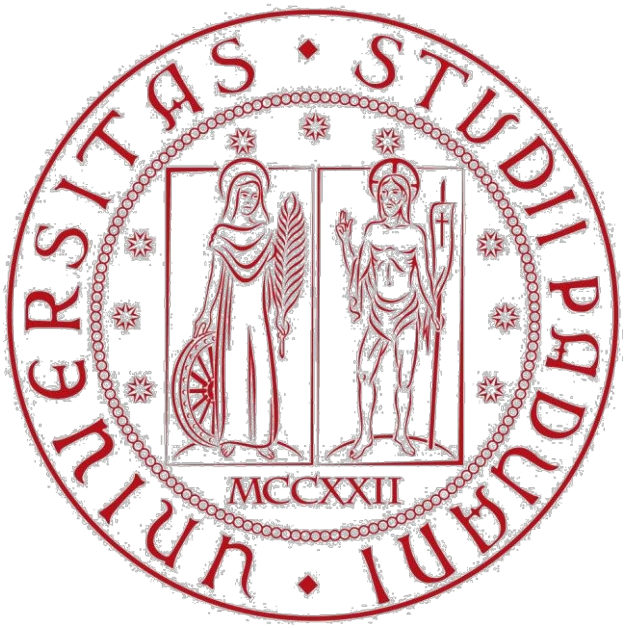


Relazione progetto per il corso di Programmazione ad Oggetti

STRUMENTO DI CALCOLO KALK

Munaro Cristiano | Matricola 1143229 | Anno 2017/2018

Scopo del progetto

Il progetto si prefigge lo scopo di realizzare un programma di calcolo Kalk che svolga attività di calcolo su tipi di dato polimorfici e non banali; Esso deve essere sviluppado in c++ nell'ambiente qt (con gui) ed in java (solo console)

Nello specifico viene realizzata una calcolatrice di espressioni logiche, aritmetiche e di funzioni

Compilazione ed esecuzione

Per la compilazione della parte c++ viene fornito oltre al codice sorgente, anche un file Kalk.pro con i comandi personalizzati di compilazione; Nello specifico si è usata una struttura di cartelle personalizzata per facilitare le modifice al model/controller view ed è abilitata la compilazione nello standard c++ 11. Per compilarlo quindi basterà lanciare i comandi:

```
qmake progetto.pro && make
```

nella root del progetto c++ (cpp)

Per compilare e lanciare il progetto Java occore lanciare all'interno della cartella java i comandi:

```
javac Use.java && java Use
```

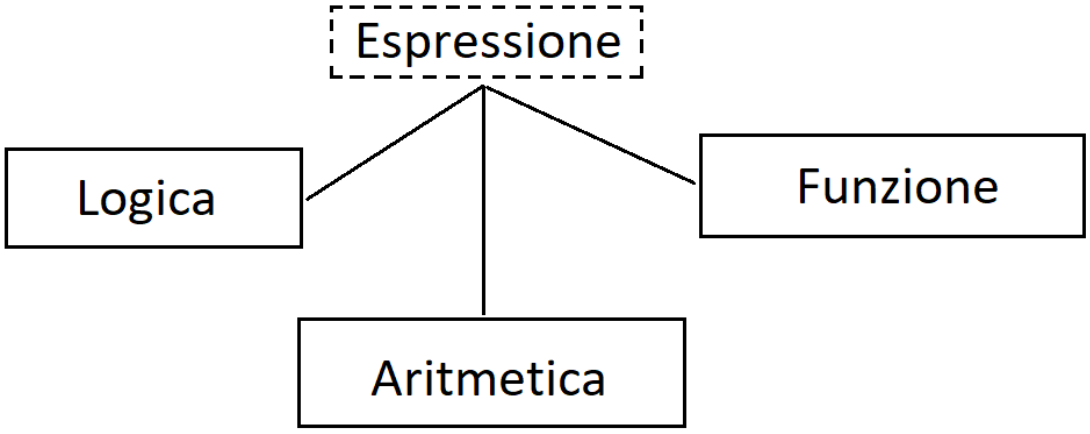
Dettagli

Nello sviluppo ho seguito il pattern MVC in modo da separare la view che genstisce i vari elementi visibili su schermo dal model, un interfaccia diretta con i tipi di dato e la logica interna della calcolatrice; Il controller è stato incorporato nella view per facilitare le comunicazioni e rendere il codice più leggibile.

Nella parte in Java ho dovuto aggiungere una classe generica Pair personalizzata per ovviare alla mancanza di std::pair.

Gerarchie dei tipi

Gerarchia delle espressioni



La classe base **Espressione** fornisce un interfaccia generica ad ogni tipo di espressione possibile ed immaginabile, è possibile infati estendere le funzionalità di Kalk con altre a piace. Essa è composta da:

Dati:

Protetti

Il testo dell'espressione

Un puntatore al risultato

Metodi:

Protetti

Costruttore per inizializzare il testo

checkParentesi per controllare se le parentesi dell'espressione siano bilanciate

pubblici

getResult che ritorna un puntatore Result (Tipo spiegato nel dettaglio in sequito)

getText per ritornare il testo dell'espressione

calculate metodo virtuale per calcolare l'espressione

ridefinizione virtuale dell'operatore di conversione a double

La classe **Logica** è una derivazione di Espressione, rappresenta un espressione logica che gestisce le operazioni AND, OR e NOT su un testo con anche altre suttoespressini logiche '(' … ')' composte da variabili e/o da costanti 0 e 1; Logica è in grado di calcolare la tabella di verità di qualunque espressione nella forma sopradescritta.

Dati:

Privati

vars vettore di coppie carattere booleano usato internamente

Metodi:

Privati

isSymbol ritorna true o folse in base al carattere passato (&, |, !, (,))

evaluate cuore della logica interna, data un espressione di immediati (1/0) congiunti da operazioni logiche, ritorna true o false in base alle regole della logica classica

Pubblici

cotruttori tramite stringa, di copia o default (si costruisce con stringa vuota)

calculate calcola la tabella di verità invocando più volte il metodo evaluate

Overload degli operatori: uguaglianza, somma, sotrazzione, decremento prefisso e conversione a double

Aritmetica rappresenta derivata di Espressione gestisce è in grado di calcolare il risultato di una serie finita di numeri e sottoespressioni congiunte da +, -, * e /

Dati:

Privati

i indice che rappresenta il punto della stringa di testo fine al quale la classe ha calcolato l'espressione. Si è deciso di mettere questo dato globale per rendere più leggibile il codice, in quanto singolo caso e condiviso da 5 metodi

Metodi:

Privati

isSymbol ritorna true o false in base al carattere passatogli (+, -, *, /, (,))

number converte il numero puntato dalla classe in intero

factor ritorna il risultato di una sottoespressione

priority ritorna il risultato di una moltiplicazione o divisione (le operazioni prioritarie)

sum ritorna il risultato di una somma o sotrazzione

Protected

evaluate metodo interno di calcolo base

isNumber analogo di isSymbol ma torna true se e solo se il carattere passatogli è un numero

Public

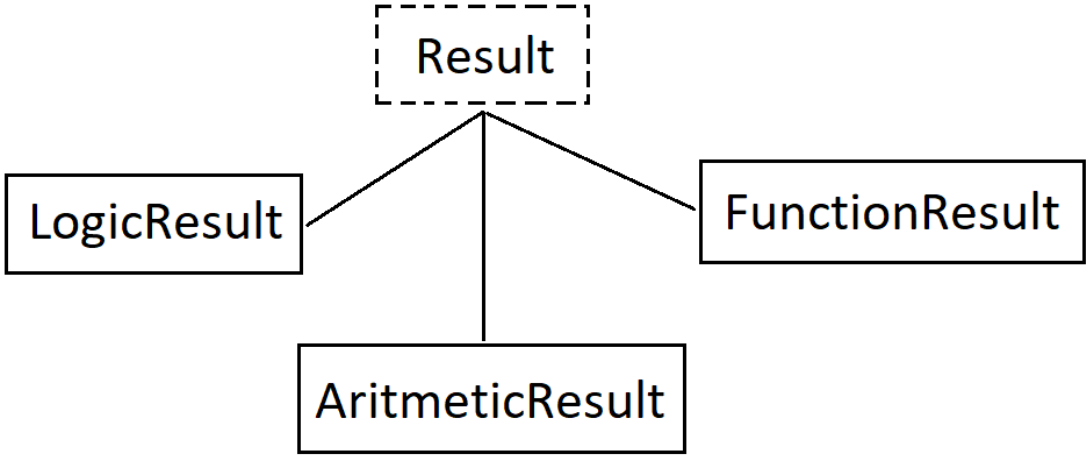
Analoghi a logica

Funzione derivata da Aritmetica è in grado di calcolare una funzione in una variabile di grado ennesimo nei numeri naturali nell'intervallo [-500, 500].

Questa classe possiede soltanto i campi dati derivati e possiede gli analoghi metodi pubblici di aritmetica, tuttavia ogni metodo è specializzato per rappresentare una funzione, non solo una espressione aritmetica. Il calcolo della funzione nell'intervallo avviene mediante l'invocazione ripetuta del metodo della classe base Aritmetica su un espressione di volta in volta diversa per calcolare la coordinata y di ogni coordinata x

Gerarchia dei risultati

Dato che I risultati delle espressioni sono diversi tra loro almeno quanto lo sono le espressioni che li generano si è resa neccessaria l'introduzione di una seconda gerarchia: I risultati



La classe astratta **Result** espone il metodo virtuale stampa

La classe **LogicalResult** rappresenta il risultato di un espressione logica

Dati:

matrix la tabella di verità

columns I nomi delle colonne

Metodi

Pubblici

Costruttori di copia e di creazione con tabella di verità e vettore nomi colonne

getColumnNames ritorna la lista ordinata dei nomi delle colonne

getMatrix ritorna la tabella di verità

stampa stampa la tabella su terminale

overload delgli operatori di assegnazione e cast to double

AritmeticResult derivata da Result rappresenta il risultato di un espressoine aritmetica

Dati:

Privati

ris double, il risultato dell'espressione

Metodi:

Pubblici

Contruttori di copia e di creazione con double

stampa stampa su console I risultati

getValue riturna il risultato come double

overload degli operatori di assegnazione e cast to double

FunctionResult derivata da Result rappresenta il risultato di una funzione

Dati:

Privati

Points coppie di punti rappresentanti la funzione nell'intervallo [-500,500]

Metodi:

Pubblici

Contruttori di copia e di creazione tramite vettore di punti

getPoints ritorna un vettori di punti

stampa stampa una tabella con le coordinate x ed y

overload degli operatori di assegnazione e di cast a double

Uso del polimorfismo

Per semplificare la lettura del codice si è usato il polimorfismo nelle gerarchie delle espressioni e dei risultati.

Una volta implementati i metodi calculate ed il cast to double in Espressione e il metodo getResult in Result è possibile usare un discreto polimorfismo per creare, calcolare e stampare il risultato di espressioni logiche, aritmetiche e pure di funzioni.

GUI

Nella gui sono stati implementati tre widget personalizzati:

Bottone, bottone dei tastierini di Kalk per consentire una facile implementazione dei pulsanti;

PlotWidget visualizzatore dei grafici in base ai punti calcolati da Funzione;

TableWidget visualizzatore delle tabelle di verità di Logica;

NOTE**:**

Cambio Tab: Nel momemto in cuoi si cambia tab, se si è appena eseguito il calcolo di un espressione, il risultato viene sommato a quello della tab in cui si sta andando.

Impegno temporale totale

Progettazione: 4-5 h

Codifica modello c++ e java / debuging: 33 h

Sviluppo interfaccia grafica qt: 12-13 h

Ambiente di Svuluppo

Sistema Operativo: OpenSUSE Tumbleweed x64

Ambiente di sviluppo C++: Qt Creator open source edition (qmake 5.5.1, g++ 5.4.0)

Ambiente di sviluppo Java: Eclipse Jupiter