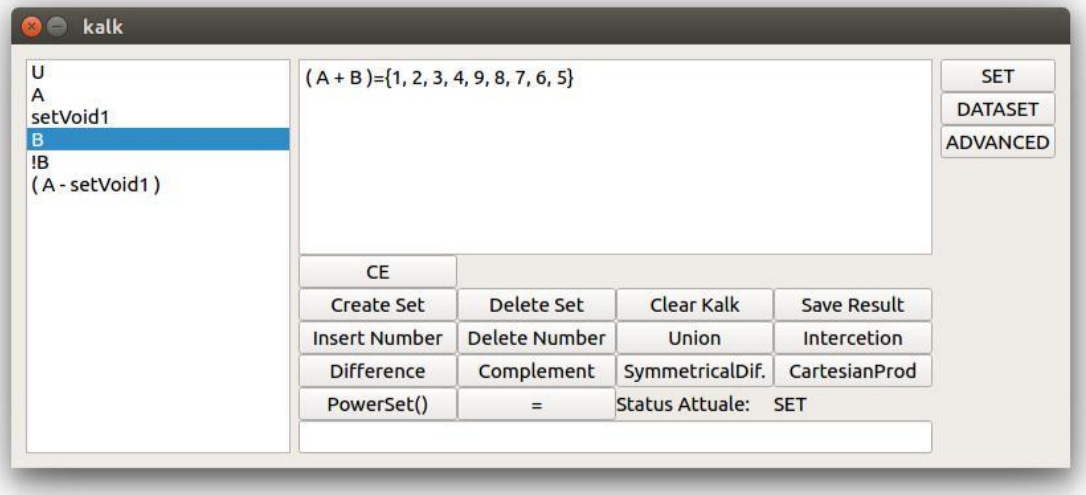
Programmazione ad oggetti
Progetto "KALK"

Relazione di Rossi Daniel
Matricola 1125444
Università degli Studi di Padova
Anno 2017/2018



Contents

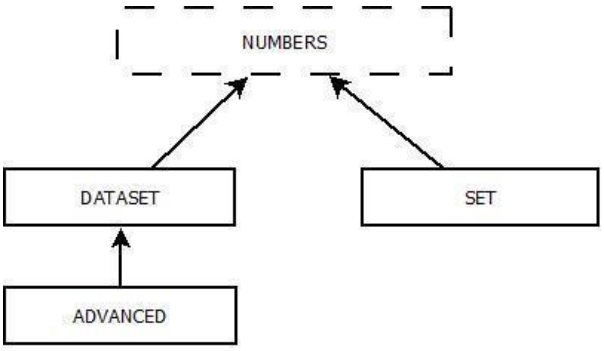
1. Scopo del progetto
2. Descrizione delle gerarchie usate 2.1. Classe Numbers
   2.2. Classe Set
   2.3. Classe Dataset
   2.4. Classe Advanced Dataset
   2.5. Classe contenitore Kalk
3. Descrizione del codice polimorfo
4. Manuale utente
5. Ore utilizzate
6. Ambiente di sviluppo

1. **Scopo del Progetto**

Il progetto si prefiggeva la realizzazione di una calcolatrice che operasse su diversi tipi di dato numerico. I dati disponibili nella calcolatrice sono: gli insiemi numerici interi detti "SET", questi permettono di eseguire le principali operazioni insiemistiche tra insiemi inseriti in un insieme universo detto"U", con la possibilità di salvarne il risultato, ogni SET conterrà come impone la teoria degli insiemi una sola istanza dello stesso valore; i DATASET comuni sono insiemi campionari di numeri interi che posso comparire anche più volte nello stesso DATASET, questi rappresentano una sequenza di misurazioni intere e forniscono le principali operazioni statistiche su un determinato insieme campionario, come media varianza e deviazione standard; gli ADVANCED dataset sono dataset con valori compresi nell'intervallo [-50 , +50], questi insiemi campionari sono pensati per essere messi in relazione tra loro per ricavare informazioni sulla correlazione dei due insiemi stessi. La calcolatrice è divisa in tre schede, una per ogni tipo di dato, ci si prefiggeva di avere schede diverse per ogni insieme.

2. **Descrizione delle gerarchie usate**

Il progetto contiene una sola gerarchia, la gerarchia dei tipi.
Questa ha alla sua base una classe astratta detta Numbers, dalla quale vengono derivate due classi concrete, Set e Dataset e da quest'ultima poi viene derivata un'ulteriore classe concreta detta Advanced. Analizziamo ora ogni singola classe:



2.1 **Classe Numbers**

La classe astratta Numbers rappresenta un'ipotetica sequenza di numeri interi.
Numbers ha un campo dati di tipo stringa che identifica ciascuna sequenza di numeri, definisce inoltre diversi metodi virtuali puri, implementati poi nelle classi derivate infine ha una classe annidata protetta "ris", utilizzata per operazioni di ricerca sempre dalle classi derivate.

2.2 **Classe Set**

La classe concreta Set, derivata dalla classe astratta Numbers, rappresenta un insieme numerico di interi univoci dove l'ordine con cui vengono inseriti non ha alcuna importanza. Implementa i metodi virtuali puri della classe base e l'overloading di alcuni operatori, rende inoltre disponibili le principali operazioni insiemistiche come l'unione, la differenza, l'intersezione ed altre.
Dà la possibilità di modificare gli insiemi sottraendo/ aggiungendo ad essi alcuni elementi, mantenendo le caratteristiche di base del Set modificato.

2.3 **Classe Dataset**

La classe concreta Dataset, derivata dalla classe astratta Numbers.
Dataset rappresenta un insieme campionario di misurazioni intere ed è caratterizzato da una sequenza di valori anche ripetuti di cui non ha alcuna importanza l'ordine. Il Dataset non può avere meno di due valori nell'insieme campionario, se si crea un Dataset vuoto o incompleto a questo verranno aggiunti tanti 0 quanti ne serviranno per ottenere una dimensione minima di 2 valori.
Implementa i metodi virtuali puri della classe base e l'overloading di alcuni operatori, inoltre rende disponibili le principali operazioni statistiche come la somma delle misurazioni, la loro media, i gradi di libertà, la varianza ed altre.
Dà la possibilità di modificare gli insiemi sottraendo/ aggiungendo ad essi alcuni elementi.

2.4 **Classe Advanced Dataset**

La classe concreta Advanced, derivata dalla classe concreta Dataset, rappresenta un insieme campionario di misurazioni intere comprese nell'intervallo intero [-50;+50].
Advanced Dataset è caratterizzato da una sequenza di valori interi anche ripetuti; è di fondamentale importanza l'ordine in cui si susseguono i valori, questo perché gli Advanced Dataset sono fatti per essere messi in relazione tra loro come valori di ascissa e ordinata di un grafico cartesiano. Inoltre dispone di alcuni campi dati di tipo decimale che contengono il risultato delle operazioni di base del Dataset come la somma delle misurazioni, la loro media, i gradi di libertà, la varianza ed altre.

La classe Advanced Dataset ridefinisce alcuni metodi della classe da cui deriva e rende disponibili operazioni tra due di essi come il coscarto, la regressione, la correlazione ed altre. Dà la possibilità di modificare gli insiemi sottraendo/ aggiungendo ad essi alcuni elementi.

Ogni Advanced ha la possibilità di essere "visto" come un Dataset, potendo eseguire su di esso tutte le operazioni concernenti il tipo Dataset, non vale però il contrario.

2.5 **Classe contenitore Kalk**

Se la gerarchia precedentemente descritta rappresenta l'implementazione dei tipi di dato su cui opera la calcolatrice, la classe Kalk rappresenta invece l'unità di controllo che gestisce la logica della calcolatrice. Kalk dispone di un campo dati che rappresenta lo status corrente del sistema, ossia su quale tipo di dato si sta operando al momento, in base a questo differenzia quali tipi di operazioni è possibile fare con i diversi tipi di dato. Dispone di una lista di puntatori polimorfi Numbers a oggetti di tipo Set, Dataset e Advanced.
Inoltre dispone di tre campi dati detti Suni, Duni, Auni, rispettivamente delle classi SetUniverse, DatasetUniverse e AdvancedUniverse, che implementano le operazioni tra tipi di dato interfacciandoli all'unità di controllo e si occupano di gestire le eccezioni che possono verificarsi per ogni tipo di dato. Sono presenti tre campi dati di tipo intero positivo (uno per tipo), che rappresentano dei contatori del numero di insiemi senza nome creati.

Sono presenti tre puntatori polimorfi di tipo Numbers, operando 1, operando 2 e risultato e un puntatore di tipo stringa ad un oggetto stringa rappresentante l'operazione che si vuole eseguire tra gli operandi scelti.
Kalk rende disponibili dei metodi che gestiscono la logica della calcolatrice.

Tutte le classi precedentemente citate fanno uso di soli costrutti della libreria STL, mantenendo così estensibilità e rendendo il codice multi piattaforma.

3. **Descrizione del codice polimorfo**

Nella gerarchia dei tipi la classe Numbers mette a disposizione i seguenti metodi virtuali:

- Per evitare memory leak il distruttore è stato reso virtuale, in modo che quando si vada ad invocare la delete sull'oggetto puntato da un puntatore di tipo Numbers venga scelto il distruttore della classe a cui l'oggetto appartiene, distruggendolo correttamente senza lasciare garbage.
- Le due funzione in e const_in, sono funzioni che fanno uso della classe annidata protetta ris, vengono utilizzate dalle classi derivate della gerarchia per verificare la presenza di un dato valore o per individuarne la posizione ritornando rispettivamente un iteratore constante e un iteratore non costante.
- Un metodo di clonazione polimorfa virtuale cosicché quando si debba assegnare un oggetto puntato ad uno degli operandi di Kalk si invochi la clone corrispondente evitando così condivisione di memoria.
- L'overloading virtualizzato dell'operatore di conversione a std::string, utile per avere una conversione dei risultati nella parte grafica evitando ridondanze nel codice.
- Un metodo virtuale Clear che inizializza l'insieme dell'oggetto di invocazione eliminando ogni elemento contenuto in esso.
- Un metodo virtuale add_value che aggiunge un elemento all'insieme di invocazione secondo le regole del tipo corrispondente.
- Un metodo virtuale sub_value che elimina un elemento dell'insieme di invocazione secondo le regole del tipo corrispondente*.
- Un metodo virtuale add_list che aggiunge una lista di elementi all'insieme di invocazione secondo le regole del tipo corrispondente.
- Un metodo virtuale sub_list che elimina una lista di elementi dall'insieme di invocazione secondo le regole dell'insieme di invocazione*.

  *Nel tipo Dataset e derivati da esso, sono stati previsti i metodi sub_value e sub_list, ma non vengono effettivamente utilizzati poiché era troppo complesso a livello di interfaccia utente indicare quale elemento si volesse eliminare.

  Si è proceduto utilizzando metodi che ad ogni modifica mostrano all'utente gli elementi attualmente presenti, gli utenti eseguono la modifica e sovrascrivono completamente i dati presenti con i nuovi dati opportunamente "parsati".

4. **Manuale utente**

All'avvio dell'applicazione Kalk questa si troverà nella scheda Set.

La calcolatrice è divisa in tre aree principali: nell'area sulla sinistra è presente una lista che in base al tipo di dato su cui si sta lavorando mostra tutti gli insiemi creati fino a quel momento, la seconda parte è a sua volta divisa in altre tre parti: nella parte superiore è presente un campo di output su cui vengono visualizzati gli insiemi selezionati e i risultati delle operazioni, nella parte centrale è presente un tastierino con una quindicina di pulsanti sui quali viene riportata una etichetta con l'operazione corrispondente; inoltre è presente una scritta che riporta lo status corrente. Infine nella parte inferiore è presente una barra di output dove verranno mostrati eventuali errori dovuti a operazioni non disponibili o a irregolarità nella costruzione di un insieme numerico.

Nella terza ed ultima parte, a destra della calcolatrice, sono presenti tre pulsanti con i quali è possibile cambiare lo status, premendo uno dei pulsanti la scheda verrà aggiornata al tipo selezionato.

La dinamica di selezione di operandi e operazioni è la stessa che in una normale calcolatrice: per effettuare operazioni tra due insiemi, si seleziona il primo insieme e successivamente l'operazione. Da questo momento non sarà più possibile cambiare il primo operando se non premendo il pulsante CE, che resetterà l'operazione corrente. Scelta l'operazione si procede scegliendo il secondo operando e premendo il pulsante =, verrà quindi stampato a schermo il risultato indicando anche l'operazione corrispondente. Nel caso di errori o irregolarità verrà riportato l'errore corrispondente nella barra inferiore.

Per operazioni di calcolo su singolo insieme, basta selezionare l'insieme su cui si vuole fare l'operazione e subito dopo l'operazione corrispondente; restituirà il risultato nella barra superiore e lascerà l'operando precedentemente selezionato inserito, cosicché se premiamo nuovamente su un'operazione singola questa verrà eseguita sullo stesso operando precedentemente scelto.

Per operazioni di manipolazione di insiemi: creazione o eliminazione di insiemi, modifica dei valori di un insieme, selezionare il pulsante corrispondente e seguire la procedura indicata nella finestra che comparirà.

**La scheda Set rende disponibile fin da subito un set chiamato "U", questo è l'insieme universo dove sarà possibile vedere tutti gli elementi inseriti in ogni Set della scheda.

Non è possibile eseguire operazioni su questo insieme se non quelle di inserimento o rimozione di valori, se si inserisce un valore questo resterà "spaiato", se si elimina un valore dall'universo verrà rimosso da tutti i Set in cui compariva.

Set rende disponibile anche una operazione di salvataggio del risultato corrente, la quale salva il risultato nella barra sulla sinistra e la rende disponibile per ulteriori operazioni.

5. **Ore utilizzate**

Progettazione: 5 ore.
Scrittura codice gerarchia tipi: 10 ore.
Scrittura codice logica di controllo: 5 ore.
Scrittura codice parte grafica: 30 ore.
Scrittura relazione: 2 ore.
Test: 2 ore.
Traduzione gerarchia tipi in Java: 2 ore.

Le forse eccessive 10 ore di costruzione della gerarchia sono dovute al fatto che non avendo chiaro inizialmente come mettere in relazione gerarchia dei tipi, unità di controllo e parte grafica ha portato a una costruzione errata della gerarchia.

6. **Ambiente di sviluppo**

Sistema operativo: Ubuntu 16.04.3 LTS
Compilatore: gcc version 5.4.0 20160609
QT: 5.9.3
QMake version 2.01a