

**Università degli studi di Padova Dipartimento di
Matematica**

Progetto di Programmazione ad Oggetti A.A. 2017-2018

Nome Progetto: Kalk

Componenti:

Nome: Damien

Cognome: Ciagola

Matricola: 1099107

Indice:

1)	Scopo del progetto.....	2
1.1)	Ambiente di sviluppo.....	2
1.2)	Ore di lavoro.....	2
2)	Descrizione gerarchia.....	3
2.1)	Descrizione codice polimorfo.....	6
3)	Manuale utente.....	7
4)	Materiale consegnato.....	11
4.1)	Parte Java.....	11

Scopo del progetto

Lo scopo del progetto KALK è lo sviluppo in C++ di una calcolatrice estendibile che permetta di calcolare le operazioni elementari e di conversione di diversi numeri in diverse basi, come prima versione troviamo le basi: (2 - 16 - 8), essa è dotata di una interfaccia utente grafica (GUI) sviluppata in Qt, adottando la filosofia del: Open-world assumption, si è cercato di separare quanto più possibile le parti logiche dalla interfaccia utente, costruendo le basi di Kalk seguendo il pattern architetturale: Model-View-Controller.

La calcolatrice è quindi in grado di gestire i calcoli di 3 diversi tipi di dati:

- 1) **Binario**
- 2) **Esadecimale**
- 3) **Ottale**

1.1 - Ambiente di sviluppo:

Sistema Operativo: Ubuntu 16.10 64-bit

- Compilatore: GCC 6.2.0
- Versione libreria Qt: 5.5.1

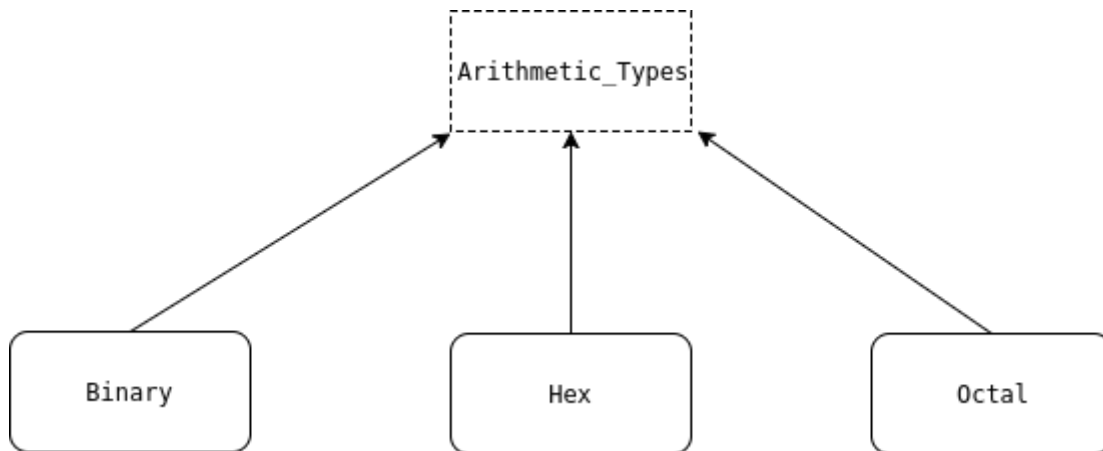
1.2 - Ore di lavoro: 70

- Modello: 10 ore (7 ore di progettazione + 3 Debug)
- Controller: 10 ore (5 ore di progettazione + 5 Debug)
- View: 20 ore (5 ore di progettazione + 15 ore di codifica)
- Correzione precedente versione: 20 ore.
- Java: 10 ore (6 ore di progettazione + 4 ore Debug)

Descrizione gerarchia

Di seguito, verranno elencate le classi e gerarchie presenti nel progetto, e le relative caratteristiche di ognuna di esse.

modello:



Arithmetic_Types: E' la classe base polimorfa astratta, che funge da interfaccia comune per le classi derivate, essa non possiede alcun campo dati.

La classe è astratta, in quanto prevede i seguenti metodi virtuali puri:

- `Arithmetic_Types* ADD(Arithmetic_Types*)`
- `Arithmetic_Types* SUB(Arithmetic_Types*)`
- `Arithmetic_Types* MUL(Arithmetic_Types*)`
- `Arithmetic_Types* DIV(Arithmetic_Types*)`
- `long double conversion_in_real() const`
- `void setNewValue(const string&)`
- `string ConvertInStringa() const`
- `double radice() const`

Inoltre sono presenti: un distruttore virtuale con comportamento di default, e una funzione statica che riceve una QString e ne fa il parser attraverso una espressione regolare, verificando se è una stringa numerica corretta in base 10, altrimenti solleva una eccezione, funzione usata in tutti i controller per garantire l'integrità delle stringhe numeriche ricevute come input.

```
virtual ~Arithmetic_Types()=default
static void parser_decimal(const QString &x)
```

- Binary, Hex, Octal:

Sono classi concrete, derivate da `Arithmetic_Types`, i cui oggetti rappresentano un numero intero in forma binaria, esadecimale e ottale.

Ogni oggetto di tipo Binary, Hex e Octal, è caratterizzato da :

- Una lista di char che rappresentano il numero del tipo scelto.

Le classi implementano i metodi virtuali puri di `Arithmetic_Types` :

- `Arithmetic_Types* ADD(Arithmetic_Types*)`
- `Arithmetic_Types* SUB(Arithmetic_Types*)`
- `Arithmetic_Types* MUL(Arithmetic_Types*)`
- `Arithmetic_Types* DIV(Arithmetic_Types*)`
- `long double conversion_in_real() const`
- `void setNewValue(const string&)`
- `string ConvertiInStringa() const`
- `double radice() const`

Contengono inoltre i seguenti metodi di classe:

- Binary:

```
static void parser(const string&)  
static Binary Converti_In_Tipo(const string&)  
static string perorsoBinaryTree(const string&)
```

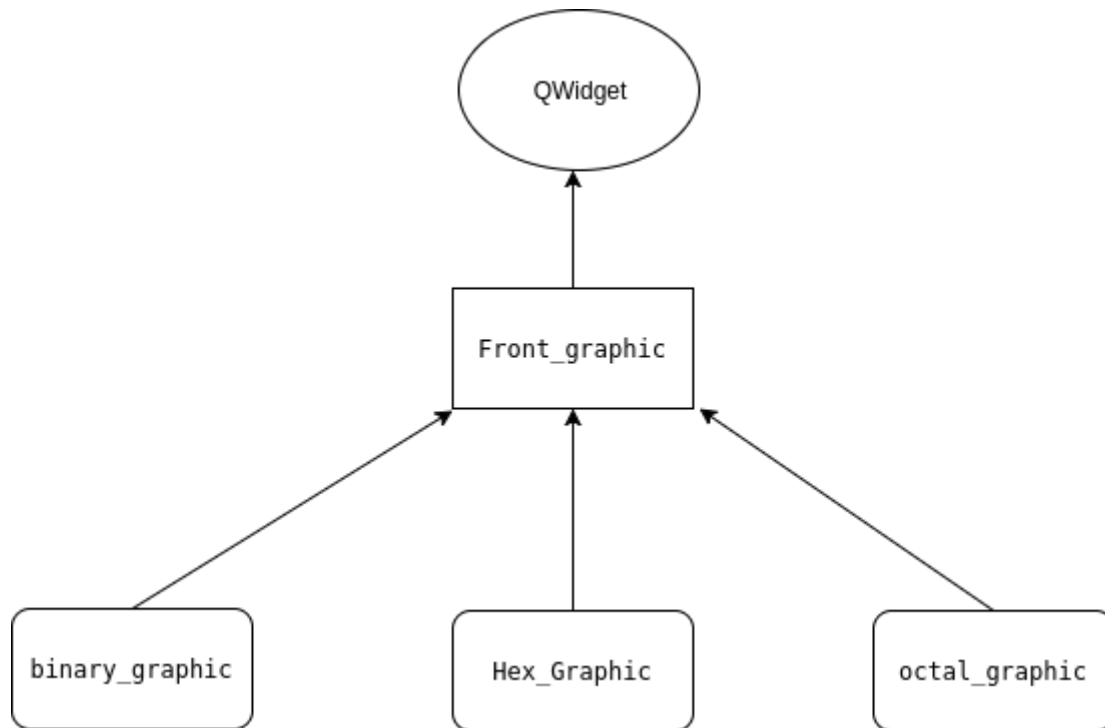
- Hex:

```
static void parser(const string&);  
static Hex Converti_In_Tipo(const string&);  
static int coverti_char_in_int(char);
```

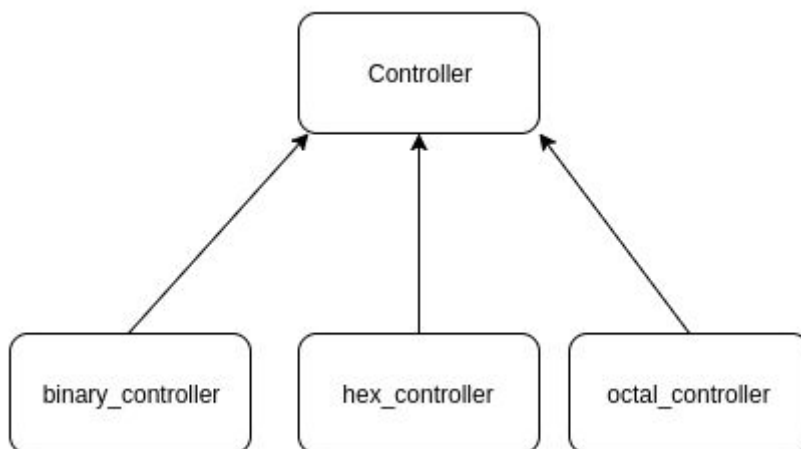
- Octal:

```
static void parser(const string&);  
static Octal Converti_In_Tipo(const string&);  
static string Charle_S_and_Emanuel_S(long);
```

View:



Controller:



Descrizione codice polimorfo

- Nella classe Calcola della Model sono presenti 3 metodi che fanno uso di codice polimorfo:

1. calcolaop1op2

è presente l'utilizzo dei metodi virtuali:

- `setNewValue(string)`
- `ADD(Arithmetic_Types*)`
- `SUB(Arithmetic_Types*)`
- `MUL(Arithmetic_Types*)`
- `DIV(Arithmetic_Types*)`
- `ConvertInStringa()`

2. calcolaRadice

è presente l'utilizzo dei metodi virtuali:

- `setNewValue(string)`
- `radice()`

3. calcolaConversioneTtoD

è presente l'utilizzo dei metodi virtuali:

- `setNewValue(string);`
- `conversion_in_real()`

Ricorrendo al late binding verranno quindi invocati i metodi sulla base del TD di op1 e op2.

Infine Grazie al distruttore virtuale di Arithmetic_Types, alla distruzione di un Arithmetic_Types viene invocato il distruttore standard della classe concreta dell' Arithmetic_Types specifico, senza così lasciare garbage nella memoria, Vedi: ~`Calcola()`.

Manuale utente

Tutti e tre i tipi permettono le operazioni elementari (+,-,*,/), il calcolo della radice quadrata, e la conversione dal tipo scelto in base10, e viceversa, Si possono quindi:

- Sommare,Sottrarre Moltiplicare e Dividere due tipi Binario/Esadecimale/Ottale
- Convertire un tipo Binario/Esadecimale/Ottale nel suo corrispettivo numero Decimale
- Convertire un numero Decimale in uno dei tre tipi sopra elencati.
- Convertire un numero Binario, Esadecimale e Ottale nel corrispettivo numero in una delle restanti basi.
- E' possibile calcolare il percorso in un albero binario.
- Calcolare il corrispettivo Colore RGB di un numero Esadecimale
- Calcolare attraverso un sistema numerico creato da Emanuel Swedenborg, un numero decimale in una stringa di caratteri usando il sistema ottale per la codifica.

Note su Emanuel.S e il sistema:

In 1716 King Charles XII of Sweden asked Emanuel Swedenborg to elaborate a number system based on 64 instead of 10. Swedenborg however argued that for people with less intelligence than the king such a big base would be too difficult and instead proposed 8 as the base. In 1718 Swedenborg wrote (but did not publish) a manuscript: "En ny rekenkonst som om vexlas wid Thetalet 8 i stället then wanliga wid Thetalet 10" ("A new arithmetic (or art of counting) which changes at the Number 8 instead of the usual at the Number 10"). The numbers 1-7 are there denoted by the consonants l, s, n, m, t, f, u (v) and zero by the vowel o. Thus 8 = "lo", 16 = "so", 24 = "no", 64 = "loo", 512 = "looo" etc.

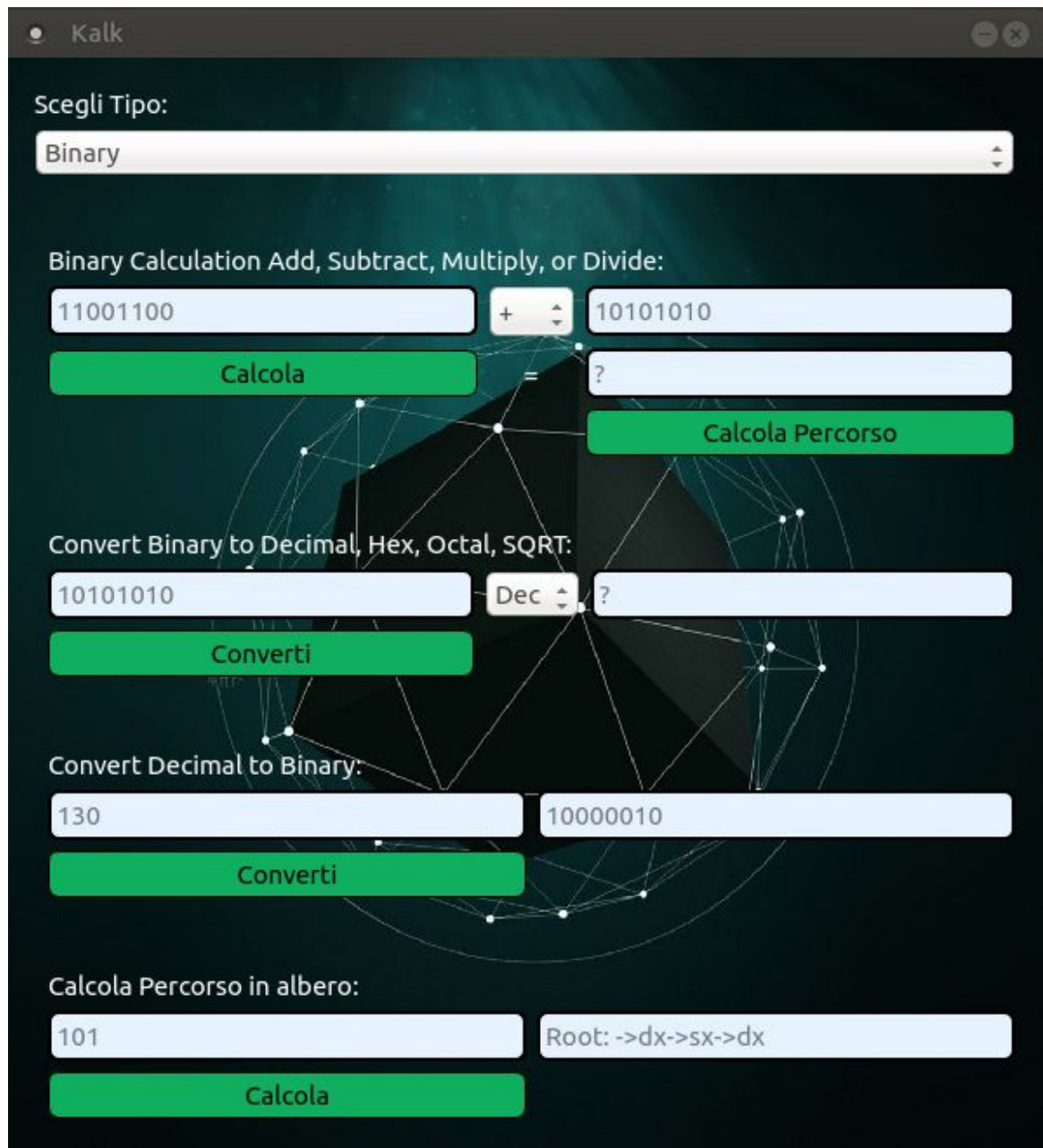
-Wikipedia from: <https://en.wikipedia.org/wiki/Octal>

La Gui generale:

Sotto la scritta "Scegli Tipo" troviamo un menù a tendina dove è possibile scegliere tra tre diversi tipi di dato (Binario,Decimale,Ottale) quando si clicca su un tipo la finestra cambia aspetto portando l'utente a visualizzare solo le operazioni consentite su quel tipo di dato.

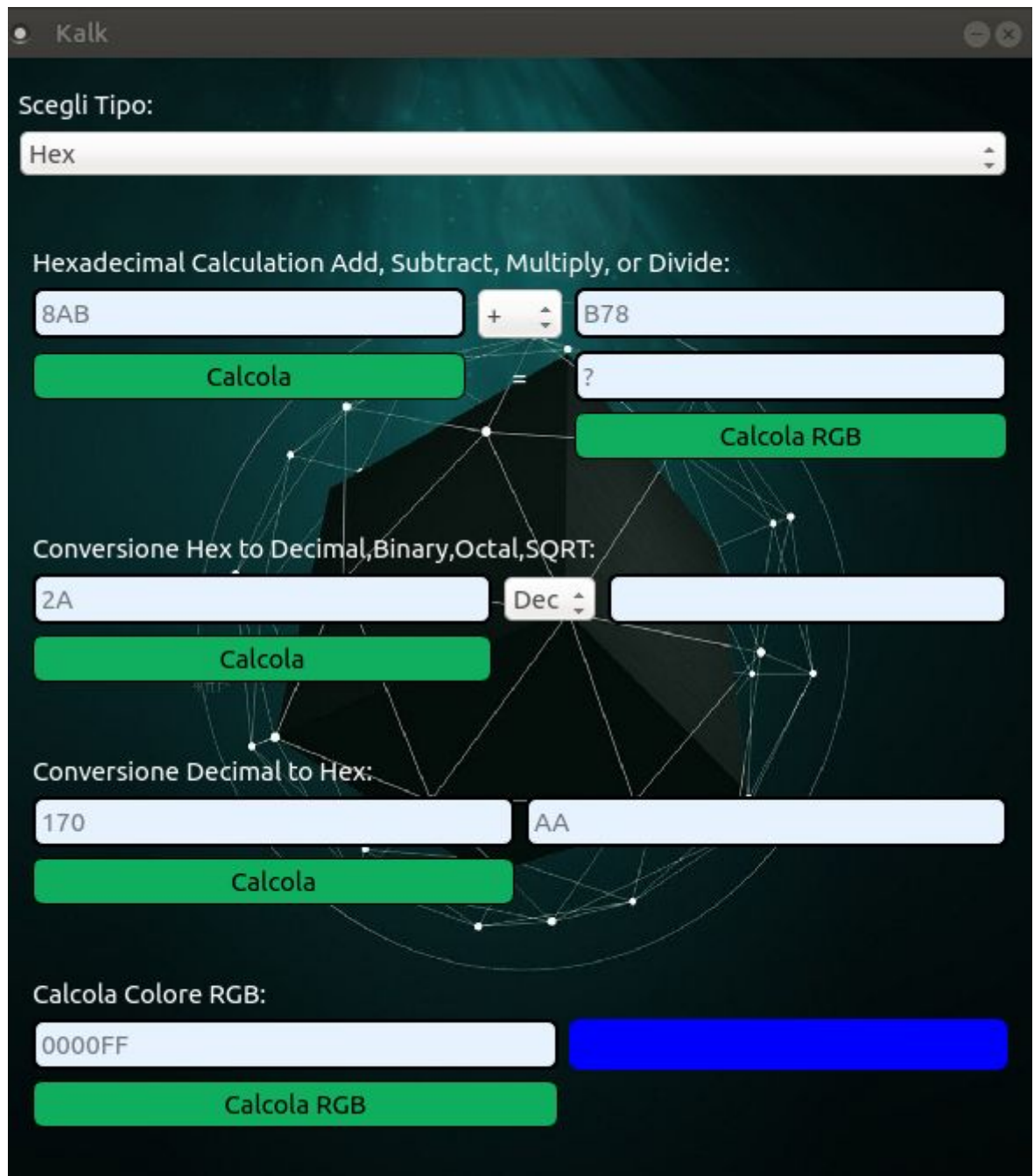
La Gui di Binary mette a disposizione:

- Binary calculator add, Subtract,Multiply or Divide:
 - ⇒ permette di calcolare (+,-,*,/) di due numeri Binari, accetta in input solo 1 o 0.
- Convert Binary to Decimal, Hex, Octal, SQRT:
 - ⇒ riceve in input sulla sinistra un numero binario, e lo converte nella base scelta dal menù a tendina, o ne calcola la radice quadrata.
- Convert Decimal to Binary:
 - ⇒Riceve in input sulla sinistra un numero decimale, e lo converte in binario.
- Calcola Percorso in albero:
 - ⇒ Riceve in input un numero binario, mostra il percorso (in un albero binario) partendo dalla radice.



La Gui di Hex mette a disposizione:

- Hexadecimal calculator add, Subtract, Multiply or Divide:
 - ⇒ permette di calcolare (+, -, *, /) di due numeri Esadecimali, accetta in input solo 1-9 e A-F.
- Convert Hex to Decimal, Binary, Octal, SQRT:
 - ⇒ riceve in input sulla sinistra un numero Esadecimale, e lo converte nella base scelta dal menù a tendina, o ne calcola la radice quadrata.
- Convert Decimal to Hex:
 - ⇒ Riceve in input sulla sinistra un numero decimale, e lo converte in Esadecimale.
- Calcola colore RGB:
 - ⇒ Riceve in input un numero esadecimale, mostra il il corrispettivo colore in RGB.



La Gui di Octal mette a disposizione:

- Octal calculator add, Subtract, Multiply or Divide:
 - ⇒ permette di calcolare (+, -, *, /) di due numeri ottali, accetta in input solo 1-7.
- Convert Octal to Decimal, Binary, Hex, SQRT:
 - ⇒ riceve in input sulla sinistra un numero ottale, e lo converte nella base scelta dal menù a tendina, o ne calcola la radice quadrata.
- Convert Decimal to Octal:
 - ⇒ Riceve in input sulla sinistra un numero decimale, e lo converte in Ottale.
- Number System Emanuel Swedenborg:
 - ⇒ Riceve in input un numero decimale, mostra la stringa di consonanti codificata tramite la base ottale.

Kalk

Scegli Tipo:

Octal

Octal Calculation Add, Subtract, Multiply, or Divide

17382 + 1734

Calcola

?

Conversione Octal to Decimal, Hex, Binary, SQRT:

8163 Dec ?

Calcola

Number System Emanuel.S

Conversione Decimal to Octal

129 201

Calcola

Number System Emanuel Swedenborg:

64 loo

Calcola

Note:

E' possibile operare solo su numeri interi, non è possibile inserire numeri Binari/Esadecimali e Ottali in virgola mobile ad esp(101.101 OR ADB.12 OR 183.15673), non è possibile convertire/sommare/sottrarre/moltiplicare/dividere un numero negativo espresso come Binario/Esadecimale/Ottale.

Materiale consegnato

- File sorgente .h e .cpp necessari per la compilazione del progetto, ordinatamente ripartiti in directory MODEL-VIEW-CONTROLLER.
- File .pro
- Kalk-(Model)-java contenente il modello del progetto scritto in Java
- relazione.pdf

Comandi per la compilazione:

\$ qmake

\$ make

\$./Kalk

Parte Java:

La parte in java è stata sviluppata in maniera molto simile a quella in C++, si è creata una classe astratta "arithmetic_types" che non contiene nessun campo dati, quindi una "Interfaccia", le funzioni di somma, sottrazione, moltiplicazione e divisione sono stati implementati attraverso dei metodi astratti (ADD,SUB,MUL,DIV), le classi della gerarchia che estendono ed implementano (attraverso un Override) i metodi di "arithmetic_types", Sono 4 (Binary,Hex,Octal,Fraction), è presente una Classe Use con un main che mostra degli esempi di uso delle rispettive classi.

Note:

Si è deciso di lasciare invariata la model della parte Java, compreso il tipo Fraction.

Ambiente di sviluppo:

Sistema Operativo: Ubuntu 16.10 64-bit

- Compilatore: javac 1.8.0
- Versione IDE: Netbeans 8.1

Comandi per la compilazione:

\$ javac *.jav

END.