Two’s Complement

- NOTE DI RILASCIO VERSIONE 1.0 -

Tecniche di programmazione

Il software è progettato secondo i paradigmi OOP con il linguaggio C++. È stato utilizzato come IDE Eclipse Oxygen.

Architettura del software

La schermata di benvenuto del programma prevede un menu che contiene le scelte disponibili ognuna delle quali dà l’accesso ad altri menu. La struttura software è così composta:

1. Convertire un numero:
   A. Decimale (intero) – Complemento a due (binario intero);
   B. Complemento a due (binario intero) – Decimale (intero);
2. Eseguire un’operazione:
   A. Operazione di cambiamento del segno
   B. Somma
   C. Sottrazione;
3. Esercitarsi con conversioni o operazioni:
   A. Conversioni:
      I. Decimale (intero) – Complemento a due (binario intero);
      II. Complemento a due (binario intero) – Decimale (intero);
   B. Operazioni:
      I. Somma;
      II. Sottrazione.
      III. Operazione di cambiamento del segno

È stata implementata una classe, Number, che contiene al suo interno tutte le funzioni necessarie alle operazioni di conversione, di somme e sottrazioni e, infine, alla modalità di esercitazione. È presente una sola funziona globale che viene chiamata prima che si crei un oggetto della classe Number ove il numero è stato implementato come una stringa solo se soddisfa certi criteri (controllati, appunto, dalla funzione globale).

Per quanto riguarda i numeri binari in complemento a due si può scegliere il numero di bit tra 4, 8 o 16.

Number

La classe Number contiene dei metodi utilizzati da tutte le sezioni del programma:

- *void parseBin()*: controlla che il numero di bit del numero binario inserito con l'utente combaci con quello precedente scelto dall'utente stesso. In caso contrario, il numero verrà opportunamente troncato o esteso a seconda dei casi.
- *void parseCa2()*: controlla che il numero di bit del numero in complemento a due inserito con l'utente combaci con quello precedente scelto dall'utente stesso. In caso contrario, il numero verrà opportunamente troncato o esteso (seguendo la logica del segno) a seconda dei casi.

Infine, è presente una funzione globale del programma prima di creare un oggetto Number:

- *bool isBinary(string op)*: controlla che il numero inserito dall'utente sia fatto solo di 0 o 1.
- *bool outOfRange(string op)*: metodo creato per controllare l'input per l'operazione unaria di cambiamento del segno; tale metodo verifica che il numero inserito non sia quello rappresentato dall'estremo inferiore di rappresentazione.

Conversioni

I metodi chiamati in questa sezione sono i seguenti (step by step):

- *string dec_ca2(int dec)*: converte in complemento a due il numero decimale intero in input;
- *int ca2_dec(string ca2)*: converte il numero in complemento a due in decimale.

Operazioni

Dopo aver verificato che il numero di bit scelto e il numero inserito siano corretti, vengono richiamati i metodi per la somma o per la sottrazione che vengono mostrate step by step. Le funzioni che riguardano questa parte implementativa sono:

- Funzioni di servizio private:
  - *string BuildStr()*: costruisce una stringa di default fatta di * quanti sono i bit degli operandi da "settare" man mano con lo svolgimento dell'operazione;
  - *void BuildRiporto(string &r, int &index)*: setta la posizione di valore index del numero r con il valore 1 per indicare che è presente il riporto;
  - *void FixPrintOutput(string op1, string op2, string rip, string sum)*: formatta l'output del risultato di un'operazione di somma;
  - *void PrintSub(string op1, string op2, string res)*: formatta l'output del risultato di una sottrazione;
  - *string Complemento()*: esegue il complemento a uno di un numero.
- Funzioni per l'overflow:
  - *bool isOverflowAdd(Number *n1, Number *n2)*: verifica se nell'operazione di somma si è verificato overflow;
  - *bool isOverflowSub(Number *n1, Number *n2)*: verifica se nell'operazione di sottrazione si è verificato overflow.
- Funzioni per l'operazione unaria di segno, la somma e la sottrazione:
  - *string bin_ca2(string bin, bool check)*: effettua l'operazione unaria di segno;
  - *friend Number& operator + (Number &n1, Number &n2)*: overloading dell'operatore + per eseguire la somma tra due numeri;
  - *friend Number& operator – (Number &n1, Number &n2)*: overloading del'operatore – per eseguire la sottrazione tra due numeri.

Esercitazioni

Conversioni

I metodi che riguardano questa sezione non sono step-by-step:

- *string dec_ca2_noTxt(int dec)*: converte il numero decimale in complemento a due;
- *int ca2_dec_noTxt(string ca2)*: il numero in complemento a due viene convertito in base 10.

Operazioni

Dopo aver scelto l'operazione e il numero di bit con il quale esercitarsi, vengono chiamate le rispettive funzioni di questa sezione. I metodi inerenti a ciò sono:

- Funzione di servizio pubbliche:
  - *string buildOp(int b)*: costruisce un numero random di bit b.
- Funzioni per le operazioni:
  - *Number& add(Number &n)*: svolge l'operazione di somma non step by step.
  - *Number& sub(Number &n)*: svolge l'operazione di sottrazione non step by step.
  - *string bin_ca2_noTxt(string bin)*: effettua l'operazione unaria di segno.

Sviluppatori e Crediti

Gabriele Marino

Per la parte implementativa ha realizzato le sezioni che riguardano le Conversioni sia step by step sia per la modalità di esercitazione. In merito alla documentazione si è occupato di stilare i paragrafi relativi alle conversioni (sia step by step sia modalità esercitazioni). Inoltre, si è occupato della parte web.

Maria Ausilia Napoli Spatafora

Per la parte implementativa ha realizzato le sezioni che riguardano le Operazioni sia step by step sia per la modalità di esercitazione. In merito alla documentazione si è occupata di stilare i paragrafi relativi alle operazioni (sia step by step sia modalità di esercitazioni), Number, note di installazione, tecniche di programmazione e architettura del software, e infine la guida utente.

Idee per successivi sviluppi

Equipaggiare il software di interfaccia GUI, inserire nuove operazioni, sostituire la scelta preliminare del numero di bit per gli operandi delle operazioni binarie con l'applicazione dell'estensione del segno all'operando di lunghezza minore.