Progetto Parte I

Corso di Programmazione 2
a.a. 2003/2004

prof. Francesco Ranzato

franz@math.unipd.it

Classe `Coppia`

Definizione di una classe `Coppia` i cui oggetti rappresentano coppie di interi (x, y).

Dovranno essere disponibili:

- costruttore di default che costruisce la coppia $(0, 0)$
- costruttore che dati due interi m e n costruisce la coppia (m, n)
- overloading di `operator<<`
- overloading di `operator==` che testa l'uguaglianza tra due coppie
- overloading di `operator!=` che testa la disuguaglianza tra due coppie

Classi `Insieme`

Definizione di una classe `Insieme_int` i cui oggetti rappresentano insiemi finiti di interi. Quindi, un oggetto di `Insieme_int` è una sequenza finita di valori di tipo `int` senza ripetizioni e in cui non conta l'ordine in cui occorrono i valori.

Inoltre, è richiesta la definizione di una classe `Insieme_Coppia`, del tutto analoga a `Insieme_int`, i cui oggetti rappresentano insiemi finiti di oggetti di `Coppia`. Basterà quindi "duplicare" il codice (purtroppo ancora non si sono visti i template che permettono di evitare ciò !).

`Insieme`: rappresentazione

È auspicabile, ma non obbligatorio, che `Insieme_`X sia una classe collezione implementata tramite la tecnica della *condivisione controllata della memoria* (cioè tramite "reference counting" e "smart pointer")

`Insieme_int`: requisiti

Dovranno essere disponibili:

- costruttore di default che costruisce l'insieme vuoto
- un metodo `bool add(int t)` che dato un intero `t` lo aggiunge all'insieme di invocazione e restituisce `true` se `t` non gli apparteneva, altrimenti, se già gli apparteneva, semplicemente restituisce `false`
- un metodo `bool remove(int t)` che dato un intero `t` lo rimuove dall'insieme di invocazione e restituisce `true` se `t` gli apparteneva, altrimenti, se non gli apparteneva, semplicemente restituisce `false`

`Insieme_int`: requisiti

- overloading di `operator==` che testa l'uguaglianza tra due insiemi
- overloading di `operator!=` che testa la disuguaglianza tra due insiemi
- overloading di `operator+` che implementa l'unione tra due insiemi
- overloading di `operator<<`
- eventuali altri costruttori, operatori o metodi utili per una corretta ed efficiente implementazione della classe

Classe `Relazione`

Definizione di una classe `Relazione` i cui oggetti rappresentano relazioni binarie su un insieme finito di interi, detto *dominio* della relazione. Quindi una relazione R su un dominio I è un insieme finito $R \subseteq I \times I$ di coppie di interi.

Sulle relazioni

Una coppia (x, y) appartenente ad una relazione R su I viene anche detta una *transizione* e y si dice un *successore* di x.

Date due relazioni $R \subseteq I \times I$ e $S \subseteq J \times J$ si definisce *composizione*, notazione RS oppure $R \circ S$, di R e S la relazione RS sul dominio $I \cup J$ definita da

$$RS \stackrel{\text{def}}{=} \{(x, z) \mid \exists y \in I.\, (x, y) \in R \text{ e } (y, z) \in S\}.$$

Quindi, RS è definita sull'unione $I \cup J$ dei due domini. In particolare, la composizione RR si denota anche con R^2, $RR^2 = R^2R$ si denota con R^3, etc.

Sulle relazioni

La relazione *inversa* di una relazione R su I è la relazione $R^{-1} \stackrel{\text{def}}{=} \{(y,x) \mid (x,y) \in R\}$ sempre sul dominio I.

La relazione $Id \stackrel{\text{def}}{=} \{(x,x) \mid x \in I$ viene detta la relazione *identità* sul dominio I.

Una relazione R su I si dice *totale* su I se per ogni $x \in I$ esiste un $y \in I$ tale che $(x,y) \in R$. L'identità su I è sempre totale su I. Se R è totale allora $Id \subseteq RR^{-1}$.

Sulle relazioni

Una relazione R si dice *transitiva* se $(x,y) \in R$ e $(y,z) \in R$ implicano $(x,z) \in R$.
Una relazione $R \subseteq I \times I$ si dice *riflessiva* se per ogni $x \in I$, $(x,x) \in R$.

La *chiusura transitiva* R^t di una relazione $R \subseteq I \times I$ è la più piccola (in senso insiemistico) relazione transitiva $S \subseteq I \times I$ che contiene R (i.e., tale che $R \subseteq S$).
La *chiusura riflessiva* R^r di una relazione R sul dominio I è la più piccola relazione riflessiva S sul dominio I che contiene R.

Sulle relazioni

Come si ottengono le chiusure transitiva e riflessiva di una relazione $R \subseteq I \times I$?

(1) Esiste un $k \in \mathbb{N}$ tale che $R^t = \bigcup_{1 \leq i \leq k} R^i$.

(2) $R^r = R \cup \{(x,x) \mid x \in I\}$.

Relazioni: esempi

$R = \varnothing$ è una relazione su qualsiasi insieme.

Siano $I = \{0,1,2,3\}$, $J = \{2,3,4\}$, $R = \{(0,1),(1,2),(3,3)\}$ su I e $S = \{(2,4),(4,4)\}$ su J. Notare che R e S non sono totali. Allora:

$RS = \{(1,4)\}$; $\quad SR = \varnothing$

$R^2 = \{(0,2),(3,3)\}$; $\quad S^2 = S$

$R^t = \{(0,1),(1,2),(3,3),(0,2)\}$

$R^r = \{(0,1),(1,2),(3,3),(0,0),(1,1),(2,2)\}$

$S^t = S$

$S^r = \{(2,4),(4,4),(2,2),(3,3)\}$

Relazione: rappresentazione

La rappresentazione per la classe `Relazione` deriva dalla definizione stessa di relazione: “una relazione consiste di un insieme di interi I che rappresenta il dominio della relazione e di un insieme di coppie di interi appartenenti a I”

Relazione: requisiti

Dovranno essere disponibili:

- costruttore di default che costruisce la relazione vuota
- costruttore a due parametri con un parametro `I` di tipo `Insieme_int` che rappresenta il dominio della relazione ed un parametro `R` di tipo `Insieme_Coppia`.
  È quindi necessario controllare che valga la condizione `R`$\subseteq$ `I`$\times$`I`
- overloading di `operator*` che ritorna la composizione di due relazioni

Relazione: requisiti

- un metodo `Relazione inversa()` che ritorna la relazione inversa della relazione di invocazione
- overloading di `operator()` con segnatura `bool operator()(int x, int y)` che ritorna `true` se e soltanto se $(\mathtt{x},\mathtt{y})$ appartiene alla relazione di invocazione, `false` altrimenti
- un metodo `Insieme_int post(int x)` che ritorna l’insieme dei successori di `x` per la relazione di invocazione

Relazione: requisiti

- un metodo `bool add(int x,int y)` che aggiunge la coppia $(\mathtt{x},\mathtt{y})$ alla relazione di invocazione e ritorna `true` se la coppia non era presente, `false` se invece era già presente. **Attenzione:** se I è il dominio della relazione di invocazione e $(\mathtt{x},\mathtt{y}) \notin I \times I$, allora anche il dominio deve essere cambiato in $I \cup \{\mathtt{x},\mathtt{y}\}$.
- un metodo `bool remove(int x, int y)` che nel caso in cui $(\mathtt{x},\mathtt{y})$ appartenga alla relazione di invocazione rimuove la coppia e ritorna `true` (il dominio non cambia), altrimenti lascia inalterata la relazione di invocazione e ritorna `false`.

`Relazione`: requisiti

- un metodo `bool total()` che ritorna `true` se la relazione di invocazione è totale, `false` altrimenti
- un metodo `void trans_closure()` che chiude per transitività la relazione di invocazione
- un metodo `void reflex_closure()` che chiude per riflessività la relazione di invocazione
- overloading di `operator<<`

Sul progetto

Nota Bene: Il presente documento va inteso come una "specifica minimale" di progetto, ossia tutto ciò che non è espressamente richiesto è appositamente lasciato a scelta libera.

Relazione: Il progetto dovrà essere accompagnato da una breve relazione scritta che descrive le principali scelte progettuali. La relazione deve essere redatta in puro formato testo (formato `.txt`).

Compilatore

Compilatore: Il progetto deve compilare (e quindi eseguire) correttamente sulle macchine Linux del laboratorio, cioè sul compilatore GNU g++ 3.2.2. È naturalmente possibile sviluppare il progetto su Windows usando il compilatore GNU g++ del sistema Cygwin in una versione successiva alla 3.2.2 (ad esempio 3.3.1).

Laboratorio

Utilizzo del laboratorio: Agli iscritti al primo compitino è stata assegnata una adeguata quota tempo per lo sviluppo del progetto.

Ricevimento in laboratorio: In prossimità della scadenza è fissato un ricevimento in laboratorio per il giorno martedì 18 novembre, ore 9:30-11:10, con i docenti per domande e consigli, anche pratici (tipicamente "perché non compila?", "perché provoca segmentation fault", etc.).

Consegna

Cosa consegnare: i file `Coppia.h`, `Coppia.cpp`, `Insieme_int.h`, `Insieme_int.cpp`, `Insieme_Coppia.h`, `Insieme_Coppia.cpp`, `Relazione.h` e `Relazione.cpp` **esattamente** con questi nomi e, per chi lo voglia, un file `main.cpp` contenente un esempio d'utilizzo. Naturalmente, se lo sviluppo dovesse richiedere altre classi allora andranno consegnati i relativi file di dichiarazione e/o definizione. Sarà messo a disposizione un esempio di `main()` minimale che dovrà compilare ed eseguire correttamente con le classi consegnate. Va inoltre consegnato il file `relazione.txt` contenente la relazione.

Consegna

Come consegnare: dalle macchine del laboratorio invocando il comando

```
consegna programmazione2-2003-2004-parte1
```

dalla directory contenente i file da consegnare. **NON** saranno accettate altre modalità di consegna (ad esempio, via email). Naturalmente è possibile consegnare remotamente il progetto (usare il server `stud56` e i comandi `ssh`, `sftp`, `scp`, etc).

Deadline: giovedì 20 novembre 2003 ore 23:59. Le consegne ritardate comporteranno una penalità secca di 1 punto sul voto finale complessivo dell'esame.