Programmazione 2

Progetto parte II (a.a. 2003/2004)

PROF. FRANCESCO RANZATO
www.math.unipd.it/∼franz
franz@math.unipd.it

1 Poset

Un *poset* (*Partially Ordered SET*) è una coppia $\mathcal{P} = \langle P, \leq \rangle$ tale che:

1. $P \subseteq$ è un insieme finito (per fissare le idee, supponiamo di caratteri, cioè `char`)
2. $\leq \subseteq P \times P$ è una relazione, per cui si usa la notazione $x \leq y$ per indicare che $(x, y) \in \leq$, detta *relazione d'ordine* su P, che soddisfa le seguenti proprietà:
   (a) riflessiva: $\forall x \in P.\ x \leq x$
   (b) transitiva: $\forall x, y, z \in P.\ (x \leq y\ \ \&\ \ y \leq z)\ \Rightarrow\ x \leq z$
   (c) antisimmetrica: $\forall x, y \in P.\ (x \leq y\ \ \&\ \ y \leq x)\ \Rightarrow\ x = y$

Quindi un poset altro non è che una particolare relazione, cioè un poset $\mathcal{P} = \langle P, \leq \rangle$ è una relazione $\leq$ sul dominio P che soddisfa le proprietà di riflessività, transitività e antisimmetria. Si noti che, per riflessività, una relazione d'ordine è sempre totale e quindi è possibile omettere il suo dominio nella definizione della relazione.

Sia $\mathcal{P} = \langle P, \leq \rangle$ un poset. Scriveremo $x < y$ quando $x \leq y$ e $x \neq y$. Se vale $x \leq y$ si dice che x è minore o uguale a y e che y è maggiore o uguale a x, mentre se vale $x < y$ si dice che x è (strettamente) minore di y e che y è (strettamente) maggiore di x.

1.1 Sulle Relazioni d'Ordine

Come ottenere una relazione d'ordine? Ricordiamo che per una relazione $R \subseteq P \times P$ qualsiasi, R^r denota la sua chiusura riflessiva e R^t denota la sua chiusura transitiva.

Fatto 1: Sia $R \subseteq P \times P$ una relazione transitiva. Allora, R è antisimmetrica se e solo se R è aciclica.

Fatto 2: Sia $R \subseteq P \times P$ una relazione qualsiasi. Allora, R è aciclica se e solo se R^t è aciclica.

Fatto 3: Sia $R \subseteq P \times P$ una relazione qualsiasi. Allora la chiusura riflessiva e transitiva R^* ($= (R^r)^t$) di R è una relazione d'ordine su P se e solo se R è aciclica.

Cosa significa che una relazione $R \subseteq P \times P$ è *aciclica*? Un ciclo in R è una sequenza $\langle x_1, ..., x_n \rangle$ di elementi in P tale che $n \geq 3$, $x_1 = x_n$, gli elementi $x_1, ..., x_{n-1}$ sono tutti distinti, e (x_1, x_2), (x_2, x_3), ..., $(x_{n-1}, x_n) \in R$. Si dice quindi che R è aciclica se e soltanto se R non contiene cicli. Quindi, per il Fatto 1, le relazioni d'ordine sono sempre acicliche. Naturalmente, non tutte le relazioni acicliche sono relazioni d'ordine.

Come testare se una relazione è ciclica? Basta testare, per ogni $x \in P$, se esiste un ciclo che include x. Per testare se esiste un ciclo che include x, inizio calcolando i successori propri di x, ovveri i successori diversi da x stesso, cioè $\mathrm{post}(x) \setminus \{x\}$. Definiamo la notazione $\mathrm{post}^-(x) = \mathrm{post}(x) \setminus \{x\}$. In generale, per

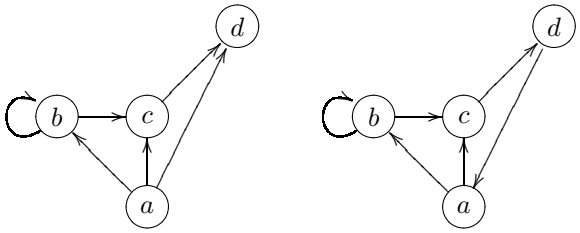

Figura 1:

un generico sottoinsieme $S \subseteq P$, usiamo inoltre la seguente notazione: $\mathrm{Fun}(S) \stackrel{\text{def}}{=} S \cup (\bigcup_{y \in S} \mathrm{post}(y))$ ($\mathrm{Fun}(S)$ è quindi l'unione di S con l'unione di tutti i post su ogni $y \in S$). Ora calcolo Fun sull'insieme dei successori propri di x, cioè $\mathrm{Fun}(\mathrm{post}^-(x))$. Se $x \in \mathrm{Fun}(\mathrm{post}^-(x))$ allora abbiamo trovato un ciclo che include x. Altrimenti, continuo calcolando $\mathrm{Fun}^2(\mathrm{post}^-(x)) = \mathrm{Fun}(\mathrm{Fun}(\mathrm{post}^-(x)))$. Se $x \in \mathrm{Fun}^2(post^-(x))$ allora ho trovato il ciclo che include x, altrimenti si continua ad iterare questo test fermandosi quando si trova un ciclo oppure quando si arriva ad un k tale che $\mathrm{Fun}^k(\mathrm{post}^-(x)) = \mathrm{Fun}^{k-1}(\mathrm{post}^-(x))$. Se trovo tale k senza aver precedentemente trovato un ciclo che include x allora non esiste nessun ciclo che include x.

Modo alternativo: riflettere sul precedente Fatto 1...

Ad esempio, in Figura 1 la relazione di sinistra R_1 è aciclica (ma non è una relazione d'ordine) mentre quella di destra R_2 è ciclica perchè contiene, ad esempio, il ciclo $\langle a, b, c, d, a\rangle$ (infatti $\mathrm{post}^-(a) = \{b, c\}$, $\mathrm{Fun}(\mathrm{post}^-(a)) = \{b, c, d\}$, $\mathrm{Fun}^2(\mathrm{post}^-(a)) = \{a, b, c, d\}$).

Il Fatto 3 induce quindi il seguente

Metodo di Generazione di Poset. Data una relazione qualsiasi $R \subseteq P \times P$ si controlla se R è aciclica. Se R è aciclica allora genero il poset $\mathcal{P} = \langle P, \leq\rangle$ dove $\leq$ è la chiusura riflessiva e transitiva di R. Altrimenti, se R è ciclica questo metodo non permette la generazione un poset.

Ad esempio, dalla relazione R_1 aciclica rappresentata in Figura 1 si ottiene una relazione d'ordine chiudendo R_1 per riflessività e transitività. Si ottiene quindi la seguente relazione d'ordine $\leq$:

$$\{a \leq a,\ a \leq b,\ a \leq c,\ a \leq d,\ b \leq b,\ b \leq c,\ b \leq d,\ c \leq c,\ c \leq d,\ d \leq d\}.$$

Vedremo in seguito un metodo per rappresentare graficamente una relazione d'ordine che ci permetterà di disegnare questa relazione d'ordine.

1.2 Massimali e Minimali

Sia $\langle P, \leq\rangle$ un poset e sia $S \subseteq P$ con $S \neq \varnothing$. L'insieme dei *massimali* di S in P è definito come segue:

$$\mathrm{Max}(S) \stackrel{\text{def}}{=} \{x \in S \mid \text{per ogni } y \in S,\ x \leq y \ \Rightarrow\ x = y\}.$$

Quindi $\mathrm{Max}(S)$ è l'insieme degli elementi "più grandi" dell'insieme S. Dualmente, l'insieme dei *minimali* di S è definito come segue:

$$\mathrm{Min}(S) \stackrel{\text{def}}{=} \{x \in S \mid \text{per ogni } y \in S,\ y \leq x \ \Rightarrow\ x = y\}.$$

Ad esempio, si consideri il poset in Figura 2. Valgono quindi:

- $\mathrm{Max}(\{a\}) = \{a\}$, $\mathrm{Max}(\{a, c\}) = \{c\}$, $\mathrm{Max}(\{a, b, c\}) = \{b, c\}$, $\mathrm{Max}(\{a, b, c, d\}) = \{b, d\}$
- $\mathrm{Min}(\{a\}) = \{a\}$, $\mathrm{Min}(\{a, c\}) = \{a\}$, $\mathrm{Min}(\{b, c\}) = \{b, c\}$, $\mathrm{Min}(\{a, b, c\}) = \{a\}$, $\mathrm{Min}(\{b, c, d\}) = \{b, c\}$, $\mathrm{Min}(\{a, b, c, d\}) = \{a\}$

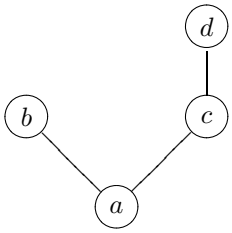

Figura 2:

1.3 Top e Bottom

Gli insiemi $\mathrm{Max}(P)$ e $\mathrm{Min}(P)$ vengono chiamati, rispettivamente, i massimali di P e i minimali di P.

Fatto 6. Per ogni poset $\langle P, \leq \rangle$ e $\varnothing \neq S \subseteq P$, $\mathrm{Max}(S) \neq \varnothing$ e $\mathrm{Min}(S) \neq \varnothing$.

Quando i massimali di P consistono di un solo elemento m, cioè $\mathrm{Max}(P) = \{m\}$, si dice che P ammette *top* e m viene detto il top (o elemento massimo) di P. Dualmente, si definisce un poset con *bottom* od elemento minimo. Ad esempio, il poset in Figura 2 ammette bottom (che è l'elemento a) ma non top in quanto $\{b, d\}$ è l'insieme dei suoi massimali.

1.4 Copertura (per la parte opzionale)

La relazione di *copertura* $\prec_{\mathcal{P}} \subseteq P \times P$ indotta da un poset $\mathcal{P} = \langle P, \leq \rangle$ è definita nel seguente modo:

$$x \prec_{\mathcal{P}} y \quad \text{se e soltanto se:}$$

1. $x < y$
2. per ogni $z \in P$, $x < z \leq y \;\Rightarrow\; z = y$

Intuitivamente, vale $x \prec_{\mathcal{P}} y$ se x è "coperto" da y, cioè se x è minore di y e non esiste nessun altro elemento z che stia tra x e y, cioè che sia maggiore di x e minore di y.

Fatto 4. La relazione di copertura indotta da un poset è sempre aciclica.

Fatto 5. Sia $\prec_{\mathcal{P}} \subseteq P \times P$ la relazione di copertura indotta da un poset $\mathcal{P} = \langle P, \leq \rangle$. Allora la chiusura riflessiva e transitiva di $\prec_{\mathcal{P}}$ coincide con la relazione d'ordine $\leq$.

In altri termini, la relazione di copertura indotta da una relazione d'ordine permette di generare per riflessività e transitività la relazione d'ordine stessa. In effetti, è anche possibile osservare che la relazione di copertura $\prec_{\mathcal{P}}$ indotta da un un poset $\mathcal{P} = \langle P, \leq \rangle$ è la più piccola relazione che permette di generare per riflessività e transitività $\leq$.

1.5 Diagrammi di Hasse (per la parte opzionale)

Le precedenti osservazioni sulle relazioni di copertura permettono di disegnare graficamente i poset tramite i cosiddetti *diagrammi di Hasse*. Si tratta di rappresentare graficamente la relazione di copertura $\prec_{\mathcal{P}}$ dal basso verso l'alto, cioè dagli elementi più piccoli agli elementi più grandi. Non si usano frecce ma semplici archi poiché si intende che gli archi siano delle frecce che vanno dal basso verso l'alto. Ad esempio, il diagramma in Figura 2 rappresenta un poset la cui relazione di copertura è $\{a \prec b,\ a \prec c,\ c \prec d\}$. Questo significa che la relazione d'ordine $\leq$ di questo poset si ottiene chiudendo per riflessività e transitività la relazione di copertura: $\{a \leq a,\ a \leq b,\ a \leq c,\ a \leq d,\ b \leq b,\ c \leq c,\ c \leq d,\ d \leq d\}$.

Consideriamo ora il poset generato dalla relazione R_1 di Figura 1. Il diagramma di questo poset è disegnato in Figura 3.

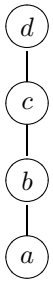

Figura 3:

1.6 Livelli (per la parte opzionale)

Sia $\langle P, \leq \rangle$ un poset. Gli elementi di P si partizionano "a livelli" seconda la seguente definizione ricorsiva:

- Livello 1: $\mathrm{Liv}_1 \stackrel{\text{def}}{=} \mathrm{Min}(P)$;
- Livello $n+1$: $\mathrm{Liv}_{n+1} \stackrel{\text{def}}{=} \{x \in P \mid \exists y \in \mathrm{Liv}_n \,.\, y \prec_{\mathcal{P}} x \text{ ed inoltre } \{z \in P \mid z \prec_{\mathcal{P}} x\} \subseteq \cup_{k \leq n} \mathrm{Liv}_k\}$.

Ad esempio, per il poset in Figura 2, abbiamo che: $\mathrm{Liv}_1 = \{a\}$, $\mathrm{Liv}_2 = \{b, c\}$, $\mathrm{Liv}_3 = \{d\}$. Se $k \in \mathbb{N}$ è il più piccolo naturale tale $\mathrm{Liv}_k \neq \varnothing$ e $\mathrm{Liv}_{k+1} = \varnothing$ allora si dice che il poset è di livello k (oppure ha k livelli). Il poset in Figura 2 ha quindi 3 livelli. Il poset vuoto è di livello 0.

Domanda: È possibile definire i livelli per una relazione qualsiasi?

Algoritmo di disegno di un poset. Per disegnare il diagramma di Hasse di un poset P si procede come segue. Si calcolano gli elementi a livello 1, cioè i minimali di P, e gli elementi di P a livello 2, e quindi si mettono gli archi dal livello 1 a questi elementi del livello 2. Successivamente, si calcolano gli elementi a livello 3 e si mettono gli archi dai precedenti elementi del livello 2 a questi elementi del livello 3. Si procede quindi in questo modo per livelli successivi sino ad arrivare al livello massimo di P. Il diagramma di Hasse è stato disegnato.

2 Catene

I poset con diagrammi di Hasse come in Figura 3 si dicono *catene*. Più precisamente, un poset $\langle P, \leq \rangle$ è una catena se la relazione d'ordine $\leq$ è *lineare*, cioè:

$$\forall x, y \in P,\ x \leq y \text{ oppure } y \leq x.$$

Naturalmente, vale che ogni catena ha top e bottom. Inoltre, se una catena ha $k > 0$ elementi allora la catena ha esattamente k livelli ed ogni livello consiste di un solo elemento.

3 Requisiti

3.1 Template `Insieme<T>`

Si richiede la definizione di una versione template comune delle classi già sviluppate `Insieme_int` e `Insieme_Coppia`, cioè un template di classe `Insieme<T>` da cui istanziare `Insieme<char>` e `Insieme<Coppia>` (notare: si intende quindi che gli oggetti di `Coppia` siano coppie di `char`; pure `Coppia` potrebbe essere un template di classe...). Naturalmente, per il template di classe restano valide le specifiche di `Insieme_int` e `Insieme_Coppia` della prima parte generalizzate ad un tipo qualsiasi `T`. Se ritenuto opportuno aggiornare il template di classe `Insieme<T>` con nuovi metodi utili.

3.2 `Relazione`

Naturalmente la classe `Relazione` va ripensata nell'ottica dell'ereditarietà: marcatore di visibilità dei campi dati e metodi, metodi virtuali, etc.

Se ritenuto opportuno aggiornare la classe `Relazione` con nuovi metodi utili. Ad esempio: è utile disporre di un generico metodo `post` con argomento un insieme di interi invece di un singolo intero? è utile disporre di un metodo che testa la proprietà di aciclicità? quali delle funzionalità richieste per `Poset` si possono già definire in `Relazione`? si possono definire i livelli in `Relazione`?

3.3 `Poset`

Definizione di una classe `Poset` derivata da `Relazione` i cui oggetti rappresentano relazioni $\leq$ su un insieme finito P tali che $\langle P, \leq \rangle$ è un poset.

Attenzione: `Poset` è quindi una `Relazione`. Il sottooggetto `Relazione` di un poset $\mathcal{P} = \langle P, \leq \rangle$ può rappresentare la relazione d'ordine completa $\leq$ oppure una relazione R che genera per chiusura riflessiva e transitiva la relazione d'ordine $\leq$ (ad esempio, questa potrebbe essere la relazione di copertura $\prec_{\mathcal{P}}$). La prima scelta è più costosa in termini di spazio ma potrebbe supportare delle implementazioni più efficienti in termini di tempo di alcuni metodi. La seconda scelta è meno onerosa in termini di spazio ma alcuni metodi potrebbero aver bisogno della relazione d'ordine completa e questa andrebbe quindi ricalcolata ogni volta. Valutare i pro e i contro in questa scelta progettuale (la soluzione più semplice è la prima...).

Dovranno essere disponibili le seguenti funzionalità:

1. costruttore di default che costruisce il poset vuoto.
2. costruttore ad un parametro `R` di tipo `Relazione` che costruisce il poset generato per riflessività e transitività dalla relazione `R` quando ciò sia possibile, cioè quando `R` è una relazione aciclica. Se `R` è ciclica, costruisce il poset vuoto.
3. **opzionale**: metodo `copertura()` che ritorna la relazione di copertura indotta dal poset di invocazione.
4. metodo `massimali(const Insieme<char>& S)` che ritorna l'insieme dei massimali di `S` nel poset di invocazione. Inoltre, il metodo duale `minimali(const Insieme<char>&)`.
5. metodi `massimali()` e `minimali()` che ritornano, rispettivamente, l'insieme dei massimali e minimali del poset di invocazione (suggerimento: usare dei valori di default nei metodi precedenti).
6. **opzionale**: un metodo `livello()` che ritorna il numero di livelli del poset di invocazione.
7. **opzionale**: un metodo `livello(int n)` che ritorna l'insieme di livello `n` del poset di invocazione. Naturalmente, quando `n` è minore o uguale a 0 oppure maggiore del numero di livelli del poset di invocazione allora ritorna l'insieme vuoto.
8. valutare la possibilità di definire un operatore di conversione esplicita a `Relazione` (dipende dal significato del sottooggetto `Relazione`, vedi sopra)
9. quando ritenuto opportuno, ridefinizioni o overriding dei metodi ereditati da `Relazione`.

3.4 `Catena`

Definizione di una classe `Catena` derivata da `Poset` i cui oggetti rappresentano relazioni $\leq$ su un insieme finito P tali che $\langle P, \leq \rangle$ è una catena.

Dovranno essere disponibili:

1. costruttore di default che costruisce la catena vuota minima (che coincide con il poset vuoto).

2. costruttore ad un parametro `R` di tipo `Relazione` che costruisce la catena generata per riflessività e transitività dalla relazione `R` quando ciò sia possibile, cioè quando `R` è una relazione aciclica e la sua chiusura riflessiva e transitiva è una catena. Quando ciò non sia possibile, costruisce la catena vuota.

3. costruttore ad un parametro `P` di tipo `Poset` che costruisce la catena dal poset `P` qualora `P` sia una catena. Quando ciò non si verifica, costruisce la catena vuota.

4. costruttore ad un parametro intero che per un intero $n \geq 0$ costruisce una catena di livello n (ad esempio $\{\texttt{'1'} \leq \texttt{'2'} \leq \texttt{'3'} \leq ... \leq \texttt{'n'}\}$), altrimenti costruisce la catena vuota.

5. metodi `top()` e `bottom()` che ritornano, rispettivamente, il top e bottom della catena di invocazione.

6. quando ritenuto opportuno, ridefinizioni o overriding dei metodi ereditati da `Relazione` o `Poset`. (Suggerimento: ad esempio, il metodo `massimali()` può essere ottimizzato sfruttando il fatto che l'insieme dei massimali di una catena consiste di un solo elemento e quindi una volta "calcolato" quell'elemento si è terminato il calcolo dei massimali.)

4 Interfaccia Grafica

Si richiede di dotare le classi sviluppate di una interfaccia grafica (GUI, Graphic User Interface) sviluppata usando la libreria wxWindows (versione 2.4.2, Linux/Windows). Le uniche richieste sulla GUI sono le seguenti:

1. La GUI mantiene sempre una relazione corrente, all'inizio non definita, cioè un oggetto il cui tipo è un sottotipo di `Relazione`.

2. La GUI mostra in una opportuna area di output la relazione corrente, specificando se si tratta di una `Relazione`, di un `Poset` o di una `Catena`.

3. La GUI deve permettere l'input per costruire oggetti e per fornire i parametri alle funzionalità disponibili. Ad esempio, per costruire una relazione l'utente specifica da una opportuna area di input un insieme R di coppie di interi che definisce una relazione il cui dominio è determinato dagli interi che appaiono in R che diventerà la relazione corrente della GUI. Si suggerisce di fare il parsing dell'input prima di effettuare le chiamate ai costruttori/metodi: se l'input passa la fase di parsing allora si effettua la corrispondente chiamata, altrimenti in caso di errore in input si avvisa l'utente dell'errore e la GUI si rimette in attesa di un nuovo evento. Per l'input si suggerisce inoltre di usare stream di stringhe. Se lo si ritiene opportuno, le funzionalità di parsing interne della GUI potrebbero essere incapsulate in una classe.

4. La GUI deve permettere:

   (a) la creazione di una relazione da input specificato dall'utente.

   (b) la creazione di un poset dalla relazione corrente, quando possibile, che diventerà quindi la nuova relazione corrente.

   (c) la creazione di una catena dalla relazione corrente (che può anche essere un poset), quando possibile, che diventerà quindi la nuova relazione corrente.

   (d) a scelta libera, creazione di relazioni, poset e catene sfruttando i relativi costruttori disponibili.

   (e) a scelta libera, il calcolo di funzionalità ritenute significative di `Relazione`, `Poset` e `Catena` sulla relazione corrente (ad esempio, `trans_closure()`, `reflex_closure()`, `post()`, `massimali()`, `minimali()`, `livello()`, `top()`, etc). L'output del calcolo della funzionalità sarà visualizzato in una opportuna area. Se la funzionalità richiede un parametro di input, questo sarà specificato dall'utente in una opportuna area di input. La richiesta minima è che la GUI renda disponibili almeno 2 funzionalità.

Lo scheletro di codice dell'interfaccia grafica fornito assieme al presente documento di specifica può essere preso come base di partenza per lo sviluppo della GUI. La libreria wxWindows è fornita assieme ad una documentazione completa e precisa che sarà la principale fonte di riferimento nello sviluppo della GUI.

4.1 Estensioni Opzionali alla GUI

L'interfaccia grafica è facilmente estendibile con ulteriori caratteristiche e funzionalità:

- la GUI rende disponibili il maggior numero di funzionalità delle classi `Relazione`, `Poset`, `Catena`.
- disegno della relazione corrente: se si tratta di una relazione allora si disegneranno delle "frecce", se si tratta di un poset (o un suo sottotipo) allora si disegneranno degli "archi". Studiare la documentazione di wxWindows per il disegno (si può disegnare su un `wxPanel`, usando un `dc` di tipo `wxClientDC` ed i metodi `SetPen()` e `DrawLine()` invocati su `dc`).
- gestione di una lista di relazioni invece che di una singola relazione che permette di rendere disponibile, ad esempio, la funzionalità di composizione di due relazioni.
- gestione dell'input/output di relazioni memorizzate su file.
- GUI curata, dettagliata e "user-friendly".
- etc.

5 Valutazione del Progetto

Un buon progetto dovrà essere sviluppato seguendo i principi basilari della programmazione orientata agli oggetti: information hiding, encapsulation, riutilizzo del codice, polimorfismo, etc, anche per quanto concerne l'interfaccia grafica. Il giudizio si baserà molto sull'aderenza del progetto a questi principi. Naturalmente le parti ed estensioni opzionali saranno valutate nel giudizio. Si ricorda inoltre che all'esame orale lo studente dovrà saper motivare le scelte progettuali e dovrà dimostrare la piena conoscenza di ogni parte del progetto. La discussione del progetto potrà portare a domande sulle nozioni fondamentali della programmazione ad oggetti in C++, istanziate al caso specifico del progetto, e le risposte dovranno dimostrare la padronanza delle conoscenze di base.

6 Regole

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.

Il progetto dovrà essere realizzato da ogni singolo studente in modo sostanzialmente **indipendente** da terze persone.

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: Il progetto deve compilare (e quindi eseguire) correttamente sulle macchine Linux del laboratorio, cioè con il compilatore GNU g++ 3.2.2 e con la libreria grafica wxWindows 2.4.2. È naturalmente possibile sviluppare il progetto su Windows usando il compilatore GNU g++ del sistema Cygwin in una versione successiva alla 3.2 e la corrispondente libreria grafica wxWindows 2.4.2 per Windows/Cygwin.

Cosa consegnare: tutti i file sorgente `.h` e `.cpp` ed, **obbligatoriamente**, un file `Makefile` per la compilazione automatizzata tramite `make`. Va inoltre consegnato il file `relazione.txt` contenente la relazione.

Come consegnare: dalle macchine del laboratorio invocando il comando

```
consegna programmazione2-2003-2004-parte2
```

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).

Scadenza di consegna: Il progetto dovrà essere consegnato rispettando **tassativamente** le scadenze previste. Di norma vale la seguente regola di scadenza: per discutere il progetto all'esame orale di un certo appello d'esame il progetto va consegnato entro il giorno (lavorativo) precedente la data dell'esame scritto di quell'appello d'esame. Naturalmente per essere ammessi all'esame orale di discussione del progetto è necessario aver riportato nella prova scritta un voto sufficiente (cioè $\geq$ 18/30). Le scadenze **ufficiali** di consegna (data e ora) verranno pubblicate nelle liste elettroniche (SIS) di iscrizione al corrispondente esame orale. Per i primi due appelli orali le scadenze sono le seguenti:

- Esame orale del 17/12/2003 ore 12:00 $\Rightarrow$ scadenza: 10/12/2003 ore 23:59.
- Esame orale del 8/1/2004 ore 10:00 $\Rightarrow$ scadenza: 29/12/2003 ore 23:59.

Per i progetti ritenuti insufficienti all'esame orale, lo studente dovrà presentarsi ad un appello orale successivo consegnando una nuova versione del progetto.