Classe polinomio

1. Introduzione e requisiti del problema
2. Specifica
3. Progetto della soluzione
4. Codifica

Requisiti del problema

Si definisca in C++ una classe Polinomio che consenta di:

- determinare il valore del polinomio in corrispondenza di un determinato valore della variabile x;
- sommare ad un polinomio un singolo termine, di cui vengono dati coefficienti e grado;
- sommare ad un polinomio un altro polinomio;
- verificare l'uguaglianza dei due polinomi;
- determinare il grado di un polinomio.

Si dia una definizione che ammetta polinomi con grado massimo pari al massimo intero rappresentabile e senza vincoli sul numero di termini del polinomio.

In questa esercitazione costruiremo una classe che gestisce l'algebra dei polinomi.

Un polinomio è costituito da una somma di termini che rappresentano potenze della variabile X, ciascuno moltiplicato per un valore intero.

La forma generale di un polinomio è:

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$$

n rappresenta il grado del polinomio.

I termini con coefficiente a_i uguale a zero, non devono essere rappresentati.

NOTA: la seguente funzione è definita in math.h:

```
double pow(double base, double esp);
elevamento a potenza di base all'esponente
exp
```

Obiettivo *dell'esercizio*

costruire una nuova struttura dati per gestire i polinomi

Funzioni richieste

- creazione polinomio
- calcolo del valore di un polinomio
- somma dei polinomi
- uguaglianza di due polinomi
- calcolo del grado di un polinomio

Non ci sono limiti al numero di termini che compongono il polinomio

↓

necessità di usare una struttura dinamica

Possibile *struttura dati* per la classe polinomio:

```
struct data {
        int grado;
        float coeff;
        data *succ;
}
```

Non ci sono limiti al numero di termini che compongono il polinomio

↓

necessità di usare una struttura dinamica

Possibile *struttura dati* per la classe polinomio:

```
struct data {
        int grado;
        float coeff;
        data *succ;
}
```

Non ci sono limiti al numero di termini che compongono il polinomio

↓

necessità di usare una struttura dinamica

Possibile *struttura dati* per la classe polinomio:

```
struct data {
        int grado;
        float coeff;
        data *succ;
}
```

Non ci sono limiti al numero di termini che compongono il polinomio

↓

necessità di usare una struttura dinamica

Possibile *struttura dati* per la classe polinomio:

```
struct data {
        int grado;
        float coeff;
        data *succ;
}
```

Una struttura dati deve essere valutata rispetto agli usi che se ne fanno e se ne faranno nel futuro.

La lista di struct è in questo caso poco efficiente.

Una struttura fortemente ricorsiva può semplificare la realizzazione di operazioni.

Gli algoritmi ricorsivi sono più semplici da realizzare rispetto agli algoritmi iterativi.
La lista di struct può non essere una soluzione efficiente.

$a_n x^n$ + Polinomio di grado < n

Struttura complicata e ricorsiva.

I programmatori che utilizzano la classe polinomio non sapranno mai come vengono immagazzinati i dati.

La scelta della struttura dati dipende esclusivamente dal progettista della classe.

- Utilizzo di una classe di supporto per la definizione della struttura dati.
- Creazione di una lista dinamica.
  (Tale lista è ordinata in senso decrescente per grado)
- La classe di supporto è definita *friend* della classe polinomio.

Definizione classe termine

Struttura dell'elemento della lista della classe termine:

- coefficiente del termine
- grado del termine
- puntatore al successivo polinomio

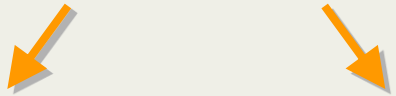
Ogni classe può avere:

- metodi per la costruzione di istanze della classe (costruttori)
- metodi per distruggere l'oggetto (distruttori)
- metodi per l'assegnamento dei valori di un oggetto della classe (metodi set)
- metodi per la restituzione dei valori di un oggetto della classe (metodi get)
- metodi specifici per la classe

Metodi classe termine

Costruttore:

costruttore per parametri di ingresso



grado *coefficiente*

Get. Calcolo del valore del termine (valore);

$coeff * valore^{grado}$

Non sono necessarie altre funzioni per la classe termine.
Tutte le operazioni di gestione della classe termine saranno compito della classe polinomio.

Metodi della classe polinomio

- Costruttore
- Distruttore
- Metodo set
- Valore
- Grado
- Somma
- =
- ==
- svuota
- stampa

Metodi della classe polinomio

- **Costruttore:** definisce il primo termine 0
- Distruttore
- Metodo set
- Valore
- Grado
- Somma
- =
- ==
- svuota
- stampa

Metodi della classe polinomio

- Costruttore
- **Distruttore:** serve per deallocare la memoria occupata dalla lista
- Metodo set
- Valore
- Grado
- Somma
- =
- ==
- svuota
- stampa

Metodi della classe polinomio

- Costruttore
- Distruttore
- **Metodo set:** aggiunta di un termine *(l'aggiunta deve essere ordinata)*
- Valore
- Grado
- Somma
- =
- ==
- Sviluppo

```
void aggiungi(int coeff, int degree)
{
  if (polinomio non nullo e degree>
      primotermine.grado )
      aggiungo un nuovo termine al polinomio
}
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- **Metodo set:** aggiunta di un termine *(l'aggiunta deve essere ordinata)*
- Valore
- Grado
- Somma
- =
- ==
- Sviluppo

```
void aggiungi(int coeff, int degree)
{
  if (polinomio non nullo e degree>
      primotermine.grado )
      aggiungo un nuovo termine al polinomio

  else if (degree< primotermine.grado)
        primotermine->next.aggiungi(coeff,degree)
}
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- **Metodo set:** aggiunta di un termine *(l'aggiunta deve essere ordinata)*
- Valore
- Grado
- Somma
- =
- ==
- Svuota

```
void aggiungi(int coeff, int degree)
{
  if (polinomio non nullo e degree>
      primotermine.grado )
      aggiungo un nuovo termine al polinomio
  else if (polinomio degree==primotermine.grado
        {
          sommo i coefficienti
          if (coefficienti==0)
                elimino primo termine
        }
  else if (degree< primotermine.grado)
        primotermine->next.aggiungi(coeff,degree)
}
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- **Metodo set:** aggiunta di un termine *(l'aggiunta deve essere ordinata)*
- Valore
- Grado
- Somma
- =
- ==
- svuota

Diverse scelte per creare un polinomio.
La scelta migliore dipende dalla funzionalità che si vuole dare alla classe.
Importanza della documentazione.

Metodi della classe polinomio

- Costruttore
- Distruttore
- Metodo set
- **Valore:**
- Grado
- Somma
- =
- ==
- svuota
- stampa

```
double valorePol(double valx) const
{
  Chiamata ricorsiva alla funzione
  valore della classe termine.
  Condizione di uscita sottopolinomio=0
  passo di ricorsione polinomio->succ.
}
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- Metodo set
- Valore
- **Grado:**
- Somma
- =
- ==
- svuota
- stampa

```
int gradoMax()
{
  Ritorna il primo termine della lista
  (che è quello con il grado più alto).
}
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- Metodo set
- Valore
- Grado
- **Somma:**
- =
- ==
- svuota
- stampa

```
void aggiungiPol(const Polinomio &altroPol)
{
  Chiamata al metodo aggiungi avente come
  parametri il coefficiente e il grado del
  primo termine del polinomio da aggiungere.
  Ricorsivamente si chiama aggiungiPol sul
  termine successivo:
  aggiungiPol(altropol->succ)
  Notate che altropol->succ è ancora
  un polinomio.
}
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- Metodo set
- Valore
- Grado
- Somma
- **=**
- ==
- svuota
- stampa

```
Polinomio operator=(const Polinomio &altroPol)
Overloading dell'operatore =
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- Metodo set
- Valore
- Grado
- Somma
- =
- ==
- svuota
- stampa

```
Boolean operator==(const Polinomio &altroPol)
 const;
  {
    Controlli sul valore di grado e
    coefficiente su tutti i termini dei
    sottopolinomi del polinomio base.
  }
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- Metodo set
- Valore
- Grado
- Somma
- =
- ==
- **svuota:**
- stampa

```
void svuota()
{
  Funzione di "servizio" che elimina tutti
  gli elementi della lista dei termini
  chiamando ricorsivamente se stessa sui
  sottopolinomi del polinomio di base.
}
```

Metodi della classe polinomio

- Costruttore
- Distruttore
- Metodo set
- Valore
- Grado
- Somma
- =
- ==
- svuota
- **stampa:**

```
void stampa() const
{
  Funzione che stampa il polinomio.
  Partendo dal primo termine si stampa il
  valore poi ricorsivamente si procede alla
  stampa dei valori dei sottopolinomi.
}
```