APPELLO SCRITTO DI PROGRAMMAZIONE 1
CORSO DI LAUREA IN MATEMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO

2014–2015

14.IX.2015

VINCENZO MARRA

Indice

Avvertenza. Il tema d'esame richiede lo sviluppo di un singolo programma che permetta di eseguire delle semplici operazioni su polinomi a coefficienti complessi. Il primo paragrafo di Preliminari fissa alcune convenzioni. Svilupperete varie funzioni ausiliarie nel risolvere gli esercizi. Integrerete poi tutte le funzioni ausiliarie in un unico programma finale, che consisterà di un unico file sorgente. (Il tema d'esame richiede esplicitamente l'implementazione di alcune funzioni ausiliarie. È comunque sempre ammissibile aggiungere altre funzioni ausiliarie qualora lo riteniate opportuno per una migliore implementazione della vostra soluzione.) Si raccomanda di leggere interamente il tema d'esame con attenzione prima di cominciare a scrivere le soluzioni dei singoli esercizi, in modo da avere chiaro l'obiettivo finale. La raccomandazione si applica a maggior ragione ai Preliminari.

Ultima revisione: 7 ottobre 2015.

Preliminari

Scriverete per questo esame un programma che permetta all'utente di eseguire alcune semplici manipolazioni di un polinomio

$$(*) \qquad c_0 + c_1 X + c_2 X^2 + \cdots + c_n X^n, \quad c_n \neq 0,$$

a coefficienti $c_i \in \mathbb{C}$ nel campo dei numeri complessi. Il *grado* del polinomio (*) è il numero intero non negativo n. Stabiliamo per convenzione che il grado del polinomio nullo (il polinomio, cioè, avente coefficienti tutti nulli) è pari a -1.

Il polinomio su cui il programma lavora è inserito dall'utente. L'utente interagisce col programma tramite un menu. Rappresenterete i numeri complessi tramite un'appropriata struttura i cui campi rappresentino la parte intera e la parte reale del numero complesso, ciascuna di tipo `double`. Rappresenterete poi il polinomio (*) tramite un array di numeri complessi, il cui elemento i-esimo contiene il coefficiente del monomio di grado i del polinomio. L'array sarà dimensionato una volta per tutte nella procedura `main` del programma. (Si noti quindi che l'array è locale alla procedura `main`, e non è invece una variabile globale.) Il numero dei suoi elementi è una costante intera `MAXEL` il cui valore, ai fini dell'esame, potete assumere pari a 10. (Potete per esempio definire la costante `MAXEL` tramite una direttiva `#define`.) Con questa convenzione, i polinomi rappresentabili hanno grado massimo pari a `MAXEL-1`.

Esercizio 1

Memorizzazione, lettura e visualizzazione di numeri complessi.

Punti: 6.

Dichiarate una struttura dati atta a memorizzare numeri complessi, rappresentati per mezzo di due `double`: la parte reale e quella immaginaria. Usate la parola chiave `typedef` per rinominare il tipo della struttura come `complex`.

Scrivete poi le tre funzioni:

- `int zeroN(complex c)`
- `void readN(...)`
- `void showN(...)`

Tutte e tre le funzioni accettano un solo parametro in ingresso, di cui dovrete stabilire il tipo laddove non già indicato. La prima funzione restituisce 0 se il valore complesso `c` in argomento è nullo, e 1 altrimenti. La seconda funzione legge dalla console un numero complesso e lo memorizza nella struttura passata in argomento, chiedendo all'utente di inserire prima la parte reale, e poi quella immaginaria. Per la lettura dei singoli `double`, usate `scanf`. La terza funzione visualizza il numero complesso nella forma $a + bi$, con a parte reale e b parte immaginaria. Cercate di scrivere la funzione in modo che la visualizzazione sia curata, nei limiti di quanto vi è possibile. Ecco alcuni suggerimenti. (I suggerimenti non sono vincolanti.)

(1) Se il numero è nullo, la funzione visualizza 0, e non per esempio $0 + 0i$ o $0i$.
(2) Se il numero è reale, la funzione visualizza a, e non per esempio $a + 0i$.
(3) Se il numero ha parte reale nulla, la funzione visualizza bi, e non per esempio $0 + bi$.
(4) Se $b < 0$, la funzione visualizza $a - |b|i$, e non per esempio $a + -|b|i$.

```
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 2
Nessun polinomio inserito.
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 1
Grado del polinomio (>= -1).
-1
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 2
0
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 3
Programma terminato.
Vincenzos-MacBook-Air:Soluzione enzo$
```

FIGURA 1. Esempio di esecuzione (Esercizio 2).

Prima di proseguire, scrivete una procedura `main` minimale per testare la vostra implementazione.

ESERCIZIO 2

Memorizzazione, lettura e visualizzazione di polinomi.

Punti: 8.

Scrivete una procedura `main` che presenti all'utente il seguente menu.

1. `Inserisci nuovo polinomio.`
2. `Visualizza polinomio corrente.`
3. `Esci.`

Se l'utente sceglie 3, il programma termina. Se l'utente sceglie 2 ma non ha ancora inserito alcun polinomio, il programma informa l'utente con un messaggio d'errore appropriato, e torna al menu. Se invece l'utente sceglie 2 e ha già inserito un polinomio, il programma visualizza il polinomio corrente e torna al menu. Se l'utente sceglie 1, il programma dà all'utente la possibilità di inserire un nuovo polinomio, che diventa quello corrente, e torna al menu. L'eventuale polinomio precedente va perduto. La lettura del polinomio avviene acquisendo prima dall'utente il grado n del polinomio da inserire (usate `scanf`), forzando il reinserimento fino a quando l'intero inserito non soddisfi $n \geq -1$. Acquisito il grado, il `main` invoca una funzione ausiliaria per leggere i coefficienti del polinomio, come dettagliato qui di seguito.

```
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 1
Grado del polinomio (>= -1).
2
Coefficiente monomio di grado 0.
Parte reale: 2.5
Parte immaginaria: -1
Coefficiente monomio di grado 1.
Parte reale: 0
Parte immaginaria: 0
Coefficiente monomio di grado (massimo) 2.
Parte reale: 0
Parte immaginaria: 0
Il coefficiente del monomio di grado massimo non puo' essere nullo.
Coefficiente monomio di grado (massimo) 2.
Parte reale: 7
Parte immaginaria: 0
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 2
2.5-1i+(7)X^2
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 3
Programma terminato.
Vincenzos-MacBook-Air:Soluzione enzo$
```

FIGURA 2. Esempio di esecuzione (Esercizio 2).

Per memorizzare il polinomio corrente il programma utilizza un array i cui elementi hanno tipo `complex`, di dimensione `MAXEL`. L'i-esimo elemento dell'array è il coefficiente del monomio di grado i. Per implementare la procedura `main` appena descritta, scrivete due funzioni ausiliarie, come segue.

La funzione

```
void readP(complex *c, int n)
```

acquisisce dall'utente un polinomio di grado `n>=-1`, e lo memorizza nell'array `c` passato in ingresso. Se `n==-1`, gli elementi dell'array sono posti tutti a zero. Altrimenti, la funzione chiede all'utente di inserire i coefficienti dei monomi fino al grado massimo, uno alla volta. La funzione controlla pure che il coefficiente del monomio di grado massimo `n` non sia nullo: se lo è, forza il reinserimento fin quando la condizione è soddisfatta.

La funzione

```
void showP(complex* c, int n)
```

visualizza il polinomio `c` di grado `n` passato in ingresso. Cercate di scrivere la funzione in modo che la visualizzazione sia curata, nei limiti di quanto vi è possibile. Si vedano le Fig. 1 e 2 per due esempi di esecuzione di una possibile risoluzione, dai quali potrete anche trarre qualche suggerimento per implementare la funzione di visualizzazione `showP`.

```
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 1
Grado del polinomio (>= -1).
3
Coefficiente monomio di grado 0.
Parte reale: 0
Parte immaginaria: 0
Coefficiente monomio di grado 1.
Parte reale: 0
Parte immaginaria: -1.5
Coefficiente monomio di grado 2.
Parte reale: -2
Parte immaginaria: 0
Coefficiente monomio di grado (massimo) 3.
Parte reale: 2.3
Parte immaginaria: -1.25
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 2
(-1.5i)X^1+(-2)X^2+(2.3-1.25i)X^3
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. Esci.
Scelta: 3
Programma terminato.
Vincenzos-MacBook-Air:Soluzione enzo$
```

FIGURA 3. Esempio di esecuzione (Esercizio 2).

ESERCIZIO 3

Due semplici test.

Punti: 7.

Aggiungete due voci al menu della procedura `main`. Il menu dovrà adesso essere:

1. `Inserisci nuovo polinomio.`
2. `Visualizza polinomio corrente.`
3. `E' un monomio?`
4. `E' reale?`
5. `Esci.`

Se l'utente sceglie 3 o 4 ma non ha ancora inserito alcun polinomio, il programma informa l'utente con un messaggio d'errore appropriato, e torna al menu. Se invece l'utente sceglie 3 e ha già inserito un polinomio, il programma verifica se il polinomio corrente è un monomio, e informa l'utente del risultato. Se l'utente sceglie 4 e ha già inserito un polinomio, il programma verifica se il polinomio corrente è reale

(ossia, ha tutti i coefficienti reali), e informa l'utente del risultato. Il polinomio nullo, di grado -1, è da considerarsi un monomio reale.

Implementate le seguenti funzioni ausiliarie.

- `int monomio(complex*, int n)` — restituisce 1 se il polinomio è un monomio, e 0 altrimenti.
- `int realeN(complex c)` — restituisce 1 se l'argomento è reale, e 0 altrimenti.
- `int realeP(complex*, int n)` — restituisce 1 se il polinomio e reale, 0 altrimenti.

Esercizio 4

Derivazione.

Punti: 9.

Aggiungete un'ultima voce al menu della procedura `main`. Il menu dovrà adesso essere

```
1. Inserisci nuovo polinomio.
2. Visualizza polinomio corrente.
3. E' un monomio?
4. E' reale?
5. Deriva.
6. Esci.
```

Se l'utente sceglie 5 ma non ha ancora inserito alcun polinomio, il programma informa l'utente con un messaggio d'errore appropriato, e torna al menu. Se invece l'utente sceglie 5 e ha già inserito un polinomio, il programma sostituisce il polinomio corrente di grado n con la sua derivata, ossia col polinomio di grado $n-1$:

$$c_1 + (2c_2)X + \cdots + (nc_n)X^{n-1}.$$

Il polinomio nullo, di grado -1, è da considerarsi la derivata di se stesso. Qualunque polinomio di grado 0 (ossia, qualunque polinomio costante) ha come derivata il polinomio nullo.

Scrivete la funzione ausiliaria

```
void deriva(complex*, int *n)
```

per implementare la derivata.

Istruzioni per la consegna

- Funzioni da consegnare (in un unico file sorgente).

  `main()`
  `zeroN()`
  `readN()`
  `showN()`
  `readP()`
  `showP()`
  `monomio()`
  `realeN`
  `deriva`
  Tutte le eventuali altre funzioni ausiliarie che abbiate scritto.

- Sito per la consegna:

  `https://upload.mat.unimi.it`

- Autenticarsi con le proprie credenziali di posta elettronica d'ateneo.
- Consegnare solo il codice sorgente (file `*.c`), non l'eseguibile.

(V. Marra) Dipartimento di Matematica *Federigo Enriques*, Università degli Studi di Milano, via Cesare Saldini, 50, I-20133 Milano
E-mail address: vincenzo.marra@unimi.it