Esame 20230728

Esercizio 2

(1) **Esercizio 2 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Secondo la formula di Taylor, la funzione seno iperbolico $\sinh(x)$ può essere approssimata dalla successione

$$\sinh_N(x) = \sum_{i=0}^{N} \frac{x^{2i+1}}{(2i+1)!}$$

in modo tale che

$$\sinh(x) = \lim_{N \rightsquigarrow \infty} \sinh_N(x)$$

Inoltre, se ϵ è l'errore massimo tollerabile, allora una possibile condizione di convergenza può essere espressa come:

$$|\sinh_N(x) - \sinh_{N-1}(x)| < \epsilon \cdot |\sinh_{N-1}(x)|$$

Scrivere nel file `esercizio2.cpp` una funzione **ricorsiva** `funzione` che prende come argomento un `float` x, un intero N, e un double ϵ e ritorna una lista concatenata dove ogni elemento in posizione $N-i$ della lista rappresenta l'approssimazione della funzione seno iperbolico $\sinh(x)$ allo step i. Ovvero, nella posizione 0 si trova il valore dell'approssimazione della funzione seno iperbolico $\sinh(x)$ che corrispondente al caso in cui si è raggiunto un errore tollerabile o si è raggiunto il valore N. L'elemento nella posizione 1, corrisponde all'approssimazione della funzion seno iperbolico $\sinh(x)$ per $N-1$, e così via. La ricorsione **termina** o quando si raggiunge N o quando per un generico risultato intermedio j la condizione $|\sinh_j(x) - \sinh_{j-1}(x)| < \epsilon \cdot |\sinh_{j-1}(x)|$ é soddisfatta.

La funzione `funzione` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). Sono **solo** consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non contengano iterazioni esplicite** (`for`, `while`, `do-while`). `Fa eccezione la funzione factorial già fornita` se si ritiene di usarla.

La funzione `funzione` è inserita in un `main` che legge dalla command line un `double` x, un opzionale intero N che rappresenta il numero massimo di iterazioni da usare nell'approssimazione di Taylor, e se N è specificato anche un opzionale ϵ che rappresenta l'errore massimo tollerato.

Di seguito sono riportati alcuni esempi di esecuzione con alcuni parametri.

```
marco > ./a.out 3.145 10 1e-20
x = 3.145
N = 10
precision = 1e-20
The list of taylor terms is: 11.588 11.588 11.588 11.588 11.588 11.588 11.58 11.497 10.894 8.3295 3.145
function(3.145) = 11.588
marco > ./a.out 3.145 20 1e-20
x = 3.145
N = 20
precision = 1e-20
The list of taylor terms is: 11.588 11.588 11.588 11.588 11.588 11.588 11.588 11.588 11.588 11.58 11.497 10.894 8.3295 3.145
function(3.145) = 11.588
marco > ./a.out 3.145 20 0.001
x = 3.145
N = 20
precision = 0.001
The list of taylor terms is: 11.58 11.497 10.894 8.3295 3.145
function(3.145) = 11.58
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `funzione`, e **caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.
- Se utile, potete utilizzare la funzione `factorial` già implementata nel file `esercizio2.cpp` (**non impatta sul vincolo delle chiamate ricorsive**), e usare `pow` da `cmath` per l'elevamento a potenza.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `cstdlib`, `ctime`, `iomanip`, e `cmath`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma.

esercizio2.cpp

Information for graders:

(2) **Esercizio 2 v2**

ESSAY | marked out of 10 | penalty 0 | File picker

Secondo la formula di Taylor, la funzione coseno iperbolico $\cosh(x)$ può essere approssimata dalla successione

$$\cosh_N(x) = \sum_{i=0}^{N} \frac{x^{2i}}{(2i)!}$$

in modo tale che

$$\cosh(x) = \lim_{N \rightsquigarrow \infty} \cosh_N(x)$$

Inoltre, se ϵ è l'errore massimo tollerabile, allora una possibile condizione di convergenza può essere espressa come:

$$|\cosh_N(x) - \cosh_{N-1}(x)| < \epsilon \cdot |\cosh_{N-1}(x)|$$

Scrivere nel file `esercizio2.cpp` una funzione **ricorsiva** `funzione` che prende come argomento un `float` x, un intero N, e un double ϵ e ritorna una lista concatenata dove ogni elemento in posizione $N - i$ della lista rappresenta l'approssimazione della funzione coseno iperbolico $\cosh(x)$ allo step i. Ovvero, nella posizione 0 si trova il valore dell'approssimazione della funzione coseno iperbolico $\cosh(x)$ che corrispondente al caso in cui si è raggiunto un errore tollerabile o si è raggiunto il valore N. L'elemento nella posizione 1, corrisponde all'approssimazione della funzion coseno iperbolico $\cosh(x)$ per $N - 1$, e così via. La ricorsione **termina** o quando si raggiunge N o quando per un generico risultato intermedio j la condizione $|\cosh_j(x) - \cosh_{j-1}(x)| < \epsilon \cdot |\cosh_{j-1}(x)|$ é soddisfatta.

La funzione `funzione` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). Sono **solo** consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non contengano iterazioni esplicite** (`for`, `while`, `do-while`). `Fa eccezione la funzione factorial già fornita` se si ritiene di usarla.

La funzione `funzione` è inserita in un `main` che legge dalla command line un `double` x, un opzionale intero N che rappresenta il numero massimo di iterazioni da usare nell'approssimazione di Taylor, e se N è specificato anche un opzionale ϵ che rappresenta l'errore massimo tollerato.

Di seguito sono riportati alcuni esempi di esecuzione con alcuni parametri.

```
marco >  ./a.out 3.145 20 1e-20
x = 3.145
N = 20
precision = 1e-20
The list of taylor terms is: 11.631 11.631 11.631 11.631 11.631 11.631 11.631 11.631 11.629 11.603 11.366 10.022 5.9455 1
function(3.145) = 11.631
marco >  ./a.out 3.145 10 1e-20
x = 3.145
N = 10
precision = 1e-20
The list of taylor terms is: 11.631 11.631 11.631 11.631 11.631 11.629 11.603 11.366 10.022 5.9455 1
function(3.145) = 11.631
marco >  ./a.out 3.145 20 0.001
x = 3.145
N = 20
precision = 0.001
The list of taylor terms is: 11.629 11.603 11.366 10.022 5.9455 1
function(3.145) = 11.629
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `funzione`, e **caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.
- Se utile, potete utilizzare la funzione `factorial` già implementata nel file `esercizio2.cpp` (**non impatta sul vincolo delle chiamate ricorsive**), e usare `pow` da `cmath` per l'elevamento a potenza.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `cstdlib`, `ctime`, `iomanip`, e `cmath`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma.

esercizio2.cpp

Information for graders:

(3) **Esercizio 2 v3**

ESSAY | marked out of 10 | penalty 0 | File picker

Secondo la formula di Taylor, la funzione seno $\sin(x)$ può essere approssimata dalla successione

$$\sin_N(x) = \sum_{i=0}^{N} (-1)^i \frac{x^{2i+1}}{(2i+1)!}$$

in modo tale che

$$\sin(x) = \lim_{N \rightsquigarrow \infty} \sin_N(x)$$

Inoltre, se ϵ è l'errore massimo tollerabile, allora una possibile condizione di convergenza può essere espressa come:

$$|\sin_N(x) - \sin_{N-1}(x)| < \epsilon \cdot |\sin_{N-1}(x)|$$

Scrivere nel file `esercizio2.cpp` una funzione **ricorsiva** `funzione` che prende come argomento un `float` x, un intero N, e un double ϵ e ritorna una lista concatenata dove ogni elemento in posizione $N - i$ della lista rappresenta l'approssimazione della funzione seno $\sin(x)$ allo step i. Ovvero, nella posizione 0 si trova il valore dell'approssimazione della funzione seno $\sin(x)$ che corrispondente al caso in cui si è raggiunto un errore tollerabile o si è raggiunto il valore N. L'elemento nella posizione 1, corrisponde all'approssimazione della funzion seno $\sin(x)$ per $N - 1$, e così via. La ricorsione **termina** o quando si raggiunge N o quando per un generico risultato intermedio j la condizione $|\sin_j(x) - \sin_{j-1}(x)| < \epsilon \cdot |\sin_{j-1}(x)|$ é soddisfatta.

La funzione `funzione` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). Sono **solo** consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non contengano iterazioni esplicite** (`for`, `while`, `do-while`). `Fa eccezione la funzione factorial già fornita` se si ritiene di usarla.

La funzione `funzione` è inserita in un `main` che legge dalla command line un `double` x, un opzionale intero N che rappresenta il numero massimo di iterazioni da usare nell'approssimazione di Taylor, e se N è specificato anche un opzionale ϵ che rappresenta l'errore massimo tollerato.

Di seguito sono riportati alcuni esempi di esecuzione con alcuni parametri.

```
marco >  ./a.out 3.145 20 1e-20
x = 3.145
N = 20
precision = 1e-20
The list of taylor terms is: -0.0034073 -0.0034073 -0.0034073 -0.0034073 -0.0034073 -0.0034073 -0.0034073 -0.0034073 -0.0034081 -0.0033859
function(3.145) = -0.0034073
marco >  ./a.out 3.145 10 1e-20
x = 3.145
N = 10
precision = 1e-20
The list of taylor terms is: -0.0034073 -0.0034073 -0.0034073 -0.0034081 -0.0033859 -0.0038588 0.0036001 -0.079351 0.52448 -2.0395 3.145
function(3.145) = -0.0034073
marco >  ./a.out 3.145 20 0.001
x = 3.145
N = 20
precision = 0.001
The list of taylor terms is: -0.0034081 -0.0033859 -0.0038588 0.0036001 -0.079351 0.52448 -2.0395 3.145
function(3.145) = -0.0034081
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `funzione`, e **caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.

- Se utile, potete utilizzare la funzione `factorial` già implementata nel file `esercizio2.cpp` (**non impatta sul vincolo delle chiamate ricorsive**), e usare `pow` da `cmath` per l'elevamento a potenza.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `cstdlib`, `ctime`, `iomanip`, e `cmath`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma.

esercizio2.cpp

Information for graders:

(4) **Esercizio 2 v4**

ESSAY | marked out of 10 | penalty 0 | File picker

Secondo la formula di Taylor, la funzione coseno $\cos(x)$ può essere approssimata dalla successione

$$\cos_N(x) = \sum_{i=0}^{N} (-1)^i \frac{x^{2i}}{(2i)!}$$

in modo tale che

$$\cos(x) = \lim_{N \rightsquigarrow \infty} \cos_N(x)$$

Inoltre, se ϵ è l'errore massimo tollerabile, allora una possibile condizione di convergenza può essere espressa come:

$$|\cos_N(x) - \cos_{N-1}(x)| < \epsilon \cdot |\cos_{N-1}(x)|$$

Scrivere nel file `esercizio2.cpp` una funzione **ricorsiva** `funzione` che prende come argomento un `float` x, un intero N, e un double ϵ e ritorna una lista concatenata dove ogni elemento in posizione $N-i$ della lista rappresenta l'approssimazione della funzione coseno $\cos(x)$ allo step i. Ovvero, nella posizione 0 si trova il valore dell'approssimazione della funzione coseno $\cos(x)$ che corrispondente al caso in cui si è raggiunto un errore tollerabile o si è raggiunto il valore N. L'elemento nella posizione 1, corrisponde all'approssimazione della funzion coseno $\cos(x)$ per $N-1$, e così via. La ricorsione **termina** o quando si raggiunge N o quando per un generico risultato intermedio j la condizione $|\cos_j(x) - \cos_{j-1}(x)| < \epsilon \cdot |\cos_{j-1}(x)|$ é soddisfatta.

La funzione `funzione` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). Sono **solo** consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non contengano iterazioni esplicite** (`for`, `while`, `do-while`). `Fa eccezione la funzione factorial già fornita` se si ritiene di usarla.

La funzione `funzione` è inserita in un `main` che legge dalla command line un `double` x, un opzionale intero N che rappresenta il numero massimo di iterazioni da usare nell'approssimazione di Taylor, e se N è specificato anche un opzionale ϵ che rappresenta l'errore massimo tollerato.

Di seguito sono riportati alcuni esempi di esecuzione con alcuni parametri.

```
marco >  ./a.out 3.145 20 1e-20
x = 3.145
N = 20
precision = 1e-20
The list of taylor terms is: -0.99999 -0.99999 -0.99999 -0.99999 -0.99999 -0.99999 -0.99999 -1 -0.99989 -1.0018 -0.97576 -1.2131 0.13084 -
function(3.145) = -0.99999
marco >  ./a.out 3.145 10 1e-20
x = 3.145
N = 10
precision = 1e-20
The list of taylor terms is: -0.99999 -0.99999 -0.99999 -1 -0.99989 -1.0018 -0.97576 -1.2131 0.13084 -3.9455 1
function(3.145) = -0.99999
marco >  ./a.out 3.145 20 0.001
x = 3.145
N = 20
precision = 0.001
The list of taylor terms is: -0.99989 -1.0018 -0.97576 -1.2131 0.13084 -3.9455 1
function(3.145) = -0.99989
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `funzione`, e **caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.

- Se utile, potete utilizzare la funzione `factorial` già implementata nel file `esercizio2.cpp` (**non impatta sul vincolo delle chiamate ricorsive**), e usare `pow` da `cmath` per l'elevamento a potenza.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `cstdlib`, `ctime`, `iomanip`, e `cmath`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma.

esercizio2.cpp

Information for graders:

Total of marks: 40