

Programmieren für Physiker

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Teilchenphysik

Prof. Dr. M. Steinhauser, Dr. A. Mildenerger
<http://comp.physik.kit.edu/>

SS 2023 – Blatt 02
Bearbeitungszeitraum: bis 03. Mai 2023

Aufgabe 4: Produkt und Doppelsumme

Pflichtaufgabe

(a) Schreiben Sie ein C++-Programm zur Berechnung des ganzzahligen Produkts

$$P_N = \prod_{i=0}^N (2i + 1) .$$

Verwenden Sie in diesem Falle zu Testzwecken nur ganzzahlige Datentypen. Lassen Sie den Benutzer N eingeben und berechnen Sie dann das Produkt, wobei Zwischenergebnisse auszugeben sind. Was passiert für große N ?

(b) Nun ist die folgende (reelle) Doppelsumme zu berechnen:

$$S_N = \sum_{i=1}^N \sum_{j=1}^i \frac{1}{i \cdot j} .$$

Nach Eingabe von N durch den Benutzer soll das Programm S_N ermitteln und ausgeben.

Aufgabe 5: Von Innen nach Außen

Pflichtaufgabe

Betrachtet wird folgender Ausdruck:

$$p_n = 1 + \underbrace{\frac{1}{3} \left(1 + \frac{2}{5} \left(1 + \frac{3}{7} \left(1 + \frac{4}{9} \left(1 + \dots \right) \right) \right) \right)}_{n \text{ Klammerpaare}}$$

Zur Erklärung: Falls im oben dargestellten Ausdruck die Punkte durch den Wert 0 ersetzt werden, ist der Ausdruck für p_4 dargestellt.

Schreiben Sie ein C++-Programm, welches nach Eingabe von n den Ausdruck p_n berechnet. Geben Sie p_n und $2p_n$ aus.

Tipp: In einer Schleife schrittweise den Ausdruck von Innen nach Außen berechnen.

Aufgabe 6: Genauigkeitstest**freiwillig**

Mithilfe eines Programmes soll untersucht werden, bis zu wie kleinen $\epsilon > 0$ der Rechner die Zahlen 1 und $1 + \epsilon$ unterscheiden kann. Für hinreichend kleine ϵ wird eine Unterscheidung nicht mehr funktionieren, da rationale Zahlen im Rechner nicht beliebig genau dargestellt sind.

Gehen Sie in Ihrem Programm folgendermaßen vor: Starten Sie mit $\epsilon = 1$. Schreiben Sie nun eine Schleife, die testet, ob 1 und $1 + \epsilon$ verschieden sind: Solange dies der Fall ist, soll innerhalb der Schleife ϵ halbiert werden. Nach der Schleife geben Sie bitte 2ϵ aus, dies ist die Größenordnung des gesuchten Ergebnisses.

Ermitteln Sie das Ergebnis für die Datentypen `float` und `double`.

Hierarchieliste der C++ Operatoren

Operator	Prio	Asso	Bedeutung
::	17		Auswahl des Bezugsrahmens (binär und unär)
[]	16	L	Indexoperator (Feldzugriff)
()		L	Funktionsaufruf und Konstruktion eines Wertes
++ --		L	Inkrement, Dekrement (Postfix: <code>x++</code> , <code>x--</code>)
.		L	Mitgliedsauswahl eines Objekts (<i>object.member</i>)
->		L	Mitgliedsauswahl bei Pointer auf ein Objekt (<i>ptr->member</i>)
!	15	R	logische Negation
~		R	bitweises Komplement
++ --		R	Inkrement, Dekrement (Prefix: <code>++x</code> , <code>--x</code>)
+ -		R	Vorzeichen
& *		R	Adressoperator (<i>&lvalue</i>), Inhaltsoperator (<i>*ptr</i>)
()		R	Typkonversion, z.B. <code>(long) x</code>
sizeof		R	Größe eines Objekts oder Datentyps
new delete		R	dynamische Speicherverwaltung
->* .*	14	L	Dereferenzierung eines Objektmitglieds
* / %	13	L	Multiplikation, Division, Modulus
+ -	12	L	Addition, Subtraktion (binär)
<< >>	11	L	bitweise Schiebe-Operatoren, auch Ein- und Ausgabe
< <= >= >	10	L	Vergleichsoperatoren kleiner, kleiner-gleich, größer-gleich, größer
== !=	9	L	Vergleich auf Gleichheit, Ungleichheit
&	8	L	bitweises Und
^	7	L	bitweises Exklusives-Oder
	6	L	bitweises Oder
&&	5	L	logisches Und
	4	L	logisches Oder
? :	3	R	bedingter Ausdruck (ternär)
=	2	R	Zuweisungsoperatoren
+= -= *= /= %=		R	Zuweisung mit Rechenoperation
>>= <<= &= ^= =		R	
,	1	L	Hintereinander-Ausführung

Erklärung der Assoziativität: `a-b-c` wird `(a-b)-c` ausgewertet, weil `'-'` zunächst links bindet. Andererseits wird z.B. `a=b=c` so ausgewertet: `a=(b=c)`. Im Zweifelsfall helfen Klammern.
