Abschlußtest Programmieren

16. Juni 2023

Name: Punkte von 32:

Gruppe: ☐ Haase ☐ Rosenberger

1. **(6 = 1+1+1+1+2 P)** Schreiben Sie die folgenden mathematischen Ausdrücke als **korrekte** C++-Anweisungen
(Variablendeklarationen, Speicherallokierungen und Fileinkludierungen sind nicht nötig, floating point Vergleiche sind erlaubt):

   (a) $a^{\frac{x^2+b}{13}}$

   (b) $\tan(\frac{\pi y}{x}) + 4y^3$ (`M_PI` ist vordefiniert)

   (c) Test, ob $|b-a| < \varepsilon \left(1 + \frac{1}{2}(|a| + |b|)\right)$

   (d) $y = \begin{cases} x+1 & \text{falls} \quad x \in [-1, 0) \\ 1-x & \text{falls} \quad x \in [0, 1] \\ 0 & \text{sonst} \end{cases}$

   (e) Gegeben sei ein `vector<int> v`. Bestimmen Sie die Anzahl aller Elemente von `v` welche positiv oder betragsmäßig größer als 11 sind.

Name: ☐ Haase ☐ Rosenberger

2. **(4 P)** Welche Ausgaben liefert das folgende Programm in den drei `cout` Anweisungen? Begründen Sie Ihre Antwort jeweils stichwortartig!

```
 1 #include <iostream>
 2 #include <vector>
 3 using namespace std;
 4
 5 vector<int> foo(vector<int> v, int &sum)
 6 {
 7   vector<int> acc;
 8   sum = -1;
 9   for (size_t k=0; k<v.size(); ++k)
10   {        if (v[k]>0)
11            {  sum += v[k];
12                    acc.push_back(v[k]);
13            }
14            else
15            {   v[k] = -v[k];  }
16   }
17
18   return acc;
19 }
20
21 int main()
22 { vector<int> v{-1,2,-4,3,1};
23   int sum(0);
24
25   auto a=foo(v,sum);
26   cout << v[0] << "::" << v[4] << endl;
27   cout << sum << endl;
28   cout << a.size() << "::"  << a[0] << endl;
29
30   return 0;
31 }
```

Name: ☐ Haase ☐ Rosenberger

3. **(4 P = 3+1)**
Finden Sie alle Paare ganzer Zahlen (x, y) aus $[-100, 100] \times [-200, 200]$ welche Lösung der diophantischen Gleichung

$$5x + 3y - 1 = 0$$

sind und geben Sie diese Lösungen aus.
Geben Sie die Anzahl der gefundenen Paare zum Schluß aus.

- Sie müssen die gefundenen Paare nicht speichern, d.h., es reicht die Ausgabe im Loop.
- Keine `break`, `continue`, `goto` oder Pointer erlaubt.

Name: ☐ Haase ☐ Rosenberger

Name: ☐ Haase ☐ Rosenberger

4. **(8 P = 3+2+2+1)**
Eine natürliche Zahl $n \in \mathbb{N}$ wird *vollkommene* Zahl (auch *perfekte* Zahl) genannt, wenn sie gleich der Summe aller ihrer (positiven) Teiler außer sich selbst ist. So ist z. B. die Zahl 6 *vollkommen*, weil 6 durch 2 und 3 teilbar ist und $1 + 2 + 3 = 6$ gilt.

   (a) Schreiben Sie eine Funktion `divisors` mit Input-Parameter n, welche alle Teiler von n außer 1 und sich selbst an das aufzurufende Programm zurückgibt.
   Hinweis: Die algorithmisch einfachste Lösung besteht im Ausprobieren aller natürlichen Zahlen $1 < d < n$.

   (b) Entwerfen Sie nun eine Funktion `perfect` mit Input-Parameter n, welche unter verwendung der Funktion `divisors` bestimmt, ob die Zahl n vollkommen ist.

   (c) Schreiben Sie schließlich eine Funktion `all_perfect`, die alle vollkommenen Zahlen $1 < n \leq N$ (N ist Input-Parameter) findet und an das aufzurufende Programm zurückgibt.

   (d) Finden Sie schließlich im Hauptprogramm (unter Verwendung der Funktion `all_perfect`) alle vollkommenen Zahlen $1 < n \leq 1000$ und geben Sie diese und deren Anzahl aus.

   Die Teile (b), (c) und (d) können auch ohne vorhandenen Code zu (a), (b) oder (c) angegeben werden. Ebenso kann (b) ohne Code zu (a) angegeben werden und (c) ohne Code zu (b) oder (a).

   Hierbei sind keine `break`, `continue`, `goto` erlaubt. Es darf in der Funktion nur **genau ein** `return`-Statement auftreten, dieses darf sich nicht innerhalb eines Zyklus oder einer Alternative befinden.
   Die Verwendung von Pointern ist nicht erlaubt.

Name: ☐ Haase ☐ Rosenberger

Name: ☐ Haase ☐ Rosenberger

5. **(10 P)** Deklarieren und definieren Sie eine Klasse `Modulo` sodaß das unten gegebene Hauptprogramm **unverändert** funktioniert. In folgenden Codezeilen werden Methoden (oder Funktionen mit) Ihrer Klasse benötigt:

   - Zeile 3: P ist die gewählte Restklasse. P muß keine Primzahl sein.
   - Zeile 4,5: Konstruktoren der Restklasse P für Zahl s.
   - Zeile 6,9,12: Ausgabe der konkreten Instanz Ihrer Restklasse.
   - Zeile 8: Addition in der Restklasse.
   - Zeile 11: Bestimmung des kleinsten Faktors $fak > 0$, sodaß in der Restklasse gilt $fak * z3 == 0$.
   - Zeile 15: Überprüfung der Kongruenz (gleiche Restklasse) einer Zahl val bzgl. $z1$. [Zusatzpunkt]
   - Default-Methoden müssen nicht deklariert/definiert werden.

   Sie können die Methodendefinitionen gleich in den -deklarationen unterbringen.
   `break`, `continue`, `goto`, und `return` in Loops/Alternativen sind **nicht erlaubt**.

```
1   int main()
2   {
3       const unsigned int P(12);
4       const Modulo z1(P,16);
5       const Modulo z2(P,15);
6       cout << z1 << endl;
7
8       const Modulo z3 = z1 + z2;              // 6 mod 5  +  7 mod 12
9       cout << z3 << endl;
10
11      unsigned int fak = z3.FactorToNull();   // Ziel: 0 mod 12 == fak * z3
12      cout << fak << " * " << z3 << "  ==  " << Modulo(P,0) << endl;
13
14      const unsigned int val(160);
15      if(z1.IsKongruent(val))
16      { cout << "Number " << val << " is element of " << z1 << endl; }
17
18      return 0;
19  }
```

Name: ☐ Haase ☐ Rosenberger