Задачи за задължителна самоподготовка

по

Обектно-ориентирано програмиране

Предефиниране на типове, функции от високо ниво, шаблони на указатели към функции. MapReduce

email: kalin@fmi.uni-sofia.bg

19 април 2017 г.

1. Инсталирайте програмата `meld` и сравнете версиите на файловете от примера за йерархията от фигури от миналата седмица и от тази седмица. Разучете как чрез `git` да получите достъп и до двете версии на файловете.

2. Като се използва шаблона `std::vector` да се създаде вектор `M` от 3 елемента, чиито елементи са вектори от по 3 числа от тип `double` (Матрица M_{3x3} с елементи от тип `double`). Да се въведат елементите на `M` от клавиатурата.

3. Да се добави нов оператор `<<` за изход в поток на `std::vector` така, че при печатане на масива от масиви (матрицата) `M` от предишната задача, елементите да се отпечатат като правоъгълна таблица - т.е. три реда, като на всеки ред има по три числа, разделени с интервали. Операторът да е универсален и да може да се ползва за всякакви двумерни матрици, построени чрез `std::vector` с елементи от всякакви (“разумни”) типове.

4. По подобие на функцията `map` за масиви, обсъдена на лекции, да се дефинира функция `map` за `std::vector` с елементи от тип `T`. Да се подготвят подходящи помощни функции така, че чрез приложение

на функцията `map` всички елементи на матрицата `M` да се увеличат с единица.

Да се напише подходящ тест.

5. Да се създаде `std::vector` от числа. Да се подготви подходяща помощна функция така, че чрез приложение на метода `map` всички елементи на масива да се отпечатат на екрана.

6. Нека е дадена следната структура `struct S {int a; int b; int c;}`. Да се дефинира и попълни примерен вектор `A` с елементи от тип `S`.

   (а) Чрез подходяща помощна функция и използване на `map`, да се отпечата сумата на полетата `a`, `b` и `c` на всеки от елементите на `A`. Да се напише подходящ тест.

   (б) Чрез подходяща помощна функция и използване на `map`, да се въведат нови стойности на висчки полета на елементите на `A` от клавиатурата.

   (в) Чрез подходяща помощна функция и използване на `map`, да се увеличат с единица всички полета `a` на елементите на `A`. Да се напише подходящ тест.

   (г) Чрез подходяща помощна функция и използване на `map`, да се разменят стойностите на *полетата* `a` и `b` на всеки от елементите на `A`. Да се напише подходящ тест.

7. По подобие на функцията `reduce` за масиви, обсъдена на лекции, да се дефинира функция `reduce` за `std::vector` с елементи от тип `T`. Чрез тазси функция и с подходящи помощни функции:

   (а) Да се намери броя на главните букви в масив от символи.

   (б) По масив от точки в равнината (`struct Point {double x,y;};`) да се определи колко от тях са в първи квадрант.

   (в) По масив от точки в равнината (`struct Point {double x,y;};`) да се определи дали всички точки лежат на правата с уравнение $5x + y + 1 = 0$.

   (г) По масив от числа да се намери най-голямото от тях.

   (д) По масив от указатели към низове (`char*`), да се намери конкатенацията им. (Внимавайте с паметта!).

Упътване:

Като първа стъпка осигурете получаване на конкатенацията без да се грижите за освобождаване на междинна памет. Операторът на `reduce` може да използва `strlen`, `strcpy` и `strcat` от `string.h`, като на всяка стъпка на конкатенацията се заделя необходимата памет. Проверете, че конкатенация се построява правилно.

След това е нужно също заделената за междинния резултат памет да се освобождава на всяка стъпка. Внимавайте с какво инициализирате началото на цикъла. Това трябва да е памет, която може да бъде освободена безопасно.

(е) По масив от коефициенти на полинома $P(x) = a_0x^n + a_1x^{n-1} + ... + a_n$, където a_i е i-тия елемент на масива, да се изчисли стойността на полинома в точката $x = 2$.

- Опитайте с помощта на *lambda* функции да изчислите стойността на полинома в няколко различни точки.

Да се напишат подходящ тестове за всяка от точките!!!

8. Да се дефинира типа на едноместни числови функции `doubleFunction = double (*) (double)`. Да се създаде вектор `std::vector<doubleFunction> functions` от едноместни числови функции.

(а) Векторът `functions` да се инициализира с *поне* 3 различни функции.

Например: $f(x) = x^2$, $g(x) = sin(x)$, $h(x) = 2x$

Можете ли да използвате *lambda* функции, за да попълните елементите на масива?

(б) Чрез използване на функцията `reduce`, да се намери най-голямата стойност измежду стойностите на всички функции в масива в точката `x=2`. Приемаме, че всички функции са дефинирани в тази точка.

- Опитайте с помощта на *lambda* функции да изчислите максималната стойност на функциите в няколко различни точки.

Упътване: Ако операторът, който подавате на `reduce` е $OP : R \times E \rightarrow R$, то типът на елементите е `doubleFunction`, а на резултата - `double`. Т.е. търсите функция:

```
double op (double crrResult, doubleFunction crrElem)
```

(в) Чрез използване на функцията `reduce`, да се намери тази (една от тези) от функциите в масива, която получава най-голяма стойност в точката `x = 2` спрямо всички функции в масива.

Упътване: Следният израз

```
reduce(functions,findMaxFun,functions[0]),
```

където `fundMaxFun` е операторът за `reduce`, който трябва да дефинирате, ще даде търсеният в условието резултат - функция. Съответно, така намерената функция можем да приложим в точката `x=2` и да отпечатаме резултата:

```
cout « reduce(functions,findMaxFun,functions[0])(2)
```

Така получената стойност ще е най-голяма измежду стойностите на всички функции в масива `functions` в точката `x=2` и ще съвпада с намерената в точка (б) стойност.

Можете ли да замените `findMaxFun` с `lambda` функция?

Напишете подходящи тестове за всяка от точките!!!