Задачи за задължителна самоподготовка по Обектно-ориентирано програмиране

Reduce, функции от високо ниво, динамична памет

email: kalin@fmi.uni-sofia.bg

8 април 2016 г.

1. Като се използва метода DynArray::reduce:
   (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.

2. Да се дефинира типа на едноместни числови функции `doubleFunction = double (*) (double)`. Да се създаде масив `DynArray<doubleFunction> functions` от едноместни числови функции.

(а) Масивът `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 метода DynArray::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 $OP : R \times E \to R$, то типът на елементите е `doubleFunction`, а на резултата - `double`. Т.е. търсите функция:

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

(в) Чрез използване на метода DynArray::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

`functions.reduce(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 « functions.reduce(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?

3. Методът DynaArray::resize, разработен на лекции, да се промени така, че при опит да се достъпи несъществуващ елемент на масива, размерът на масива да нараства поне два пъти но само при условие, че всяка от последните 3 операции за достъп е наложила увеличаване на размера на масива. В противен случай, масивът да нараства само с минималния необходим капацитет.

Пример:

```
DynArray<int> da(1);
da[0] = 0; //no resize
da[1] = 1; //increase capacity with 1
da[2] = 2; //increase capacity with 1
da[2] = 0; //no resize
da[3] = 3; //increase capacity with 1
           //(last operation did not require
              resize)
da[4] = 4; //increase capacity with 1
da[5] = 5; //increase capacity with 1
da[6] = 6; //double capacity
           //(after three subsequent resizings)
```

4. Класът DynaArray, разработен на лекции, да се разшири с оператор `*`, който построява "сечението" на два масива. Масивът `C` наричаме сечение на масивите `A` и `B`, ако в `C` няма повторения на елементите и `C` се състои точно от тези елементи, който са елементи както на `A`, така и на `B`. Пример: `[1,2,2,5,3]*[2,3,4]=[2,3]`.

- Да се дефинира и съответният оператор `*=`.