АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА НЕНАМАЛЯВАЩА РЕДИЦА

Първи начин.

В условието няма ограничение за броя на числата в редицата, затова няма да използваме масив.

Отначало прочитаме първите две числа в променливите **x** и **y**. В променливата **x** ще поддържаме предпоследното прочетено число, а в променливата **y** – последното прочетено число. Четенето продължава, докато стойността на **y** е положителна.

```
cin >> x;
cin >> y;
while(y>0)
{ // Обработка
  // . . .
  // . . .

  x = y;
  cin >> y;
}
```

В променлива **k** поддържаме дължината на текущата ненамаляваща редица, а в променливата **kmax** – дължината на най-дългата открита до момента ненамаляваща редица. Ако **x<=y,** увеличаваме **k** с единица. Когато **x>y,** текущата редица завършва с **x** и от **y** започва нова редица, която отначало е с дължина 1. Преди това, обаче, трябва да проверим дали завършилата редица не е по-дълга от **kmax** и ако е така, да обновим стойността на **kmax**. Не трябва да забравяме накрая да проверим дали най-дългата ненамаляваща подредица не се намира в края на дадената редица.

Втори начин

Този начин предполага съхраняване на всички числа от редицата в паметта на компютъра. Тъй като не е зададено ограничение за техния брой, то за съхраняване на числата от редицата използваме вектор. Това решение е по-скоро за преподавателите, които могат да го използват като повод да въведат учениците си в Стандартната библиотека на C++ и използването на вектори.

```
vector<double> a;

double x;
cin >> x;
while(x>0)
{ a.push_back(x);
  cin >> x;
}

int n = a.size();
```

Сега числата от дадената редица са в елементите

a[0],a[1],a[2],...,a[n-1].

За i=1,2,...,n-1 проверяваме дали числото a[i] продължава текущата ненамаляваща подредица или е начало на нова подредица.

```
for(int i=1; i<n; i++)
  if(a[i-1]<=a[i])
  { k++;
    if(k>kmax) kmax=k;
  }
```

Автор: Донка Даракчиева