1. Напишете функция optimum () с прототип:

int optimum(int arr[], int n, bool (\*fp)(int, int));

която връща оптимума – минималният или максималният елемент на масива в зависимост от подавания указател към функция. За повикването на optimum() използвайте разгледаните на лекцията

bool compare\_less(int a, int b) {

return a < b;

}

bool compare\_greater(int a, int b) {

return a > b;

}

или съответстващите им ламбда-изрази.

Покажете използването на фунцията optimum() за намиране на минималния и максималния елемент на масив.

*Подсказка*. Това е типовата задача за намиране на минимум (максимум) на масив. На резултата (нека е с име result) на функцията optimum() първоначално се присвоява първият елемент на масива-параметър. Следва цикъл, в който се сравнява текущата стойност на оптимума с текущия елемент на масива посредством функцията, подадена на указателя fp. Aко резултатът е true, стойността на result се сменя. Накрая функцията връща result.

1. Реализирайте optimum (), compare\_less() и compare\_greater() oт задача 1 **като шаблони на функции** и покажете използването им за масиви от тип int, double и string.

Тъй като още не сме разглеждали класа string, масив от тип string декларираме примерно така:

string cities[4] {“Burgas”, “Varna”, “Sofia”, “Ruse”};

Подсказка: За целта преди всяка функция запишете

template<typename T>

а типа int (oсвен int n) заменете на параметъра-тип T.

1. Като помощна функция напишете шаблон на функция print(), която извежда масив от произволен тип (това е задача oт ДР5).

В main() декларирайте инициализирани масиви от тип double, int и string.

Сортирайте ги с помощта на функцията sort() от STL по нарастващ и по намаляващ ред , като за втория вариант използвате ламбда израз. Преди и след всяко сортиране използвайте функцията print() за извеждане на резултата.

1. Дефинирайте структура Worker(работник) съдържаща полетата:

* име на работника
* заемана длъжност (позиция)
* заплата

Напишете функции

* за въвеждане на променлива Worker от конзолата

void inputWorker(Worker& w) *//оverload 1*

void inputWorker(Worker\* w) *//оverload 2*

* за извеждане на променлива Worker в конзолата

void printWorker(Worker w)

* за извеждане на масив Worker в конзолата

void printWorkers(Worker w[], int count)

* за извеждане на работниците, заемащи определена позиция, в конзолата

void printWorkersGivenPosition(Worker w[], int count, string position)

Във функцията main()

* Декларирайте неинициализирана променлива Worker
* Въведете елементите й с помощта на фунцията inputWorker (оverload 1)
* Изведете я на екрана
* Декларирайте инициализирана променлива Worker
* Изведете я на екрана
* Декларирайте указател към неинициализирана променлива Worker, разположена в динамичната памет
* Въведете елементите на сочената от указателя структурна променлива с помощта на фунцията inputWorker (оverload 2)
* Изведете я на екрана
* Декларирайте указател към инициализирана променлива Worker, разположена в динамичната памет
* Изведете я на екрана
* Декларирайте масив workers от 4 елемента от тип Worker, инициализиран
* изведете таблица с данните за работниците - масива workers ( извикайте printWorkers())
* изчислете и изведете на екрана средната заплата на служителите (използвайте range based for)
* въведете от конзолата позицията, за която искате да направите справка
* извикайте функцията printWorkersGivenPosition(), която да изведе работниците на тази позиция.

Допълнение за реализацията на printWorkersGivenPosition(): т.к. функцията e void, може в самата функция да използвате спомагателна променлива от тип bool с начална стойност false, която да се промени на true, ако е намерен работник на дадената позиция, или от тип int, която да преброи съвпаденията за позицията. Идеята е ако не е намерен работник на търсената позиция, вместо списъка с работници на търсената позиция функцията да изведе съобщение, че няма работник(ци) на тази позиция.