Dostupná místa

Termín odevzdání:	**29.04.2018 23:59:59**
Pozdní odevzdání s penalizací:	**01.07.2018 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**6.0000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	3 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je realizovat šablonu třídy, která dokáže vyhledávat propojená místa.

Předpokládáme mapu, ve které se nacházejí místa (uzly) a spojnice mezi uzly (spojení). Pod místem si lze představit města na mapě, adresy počítačů v síti, ... Spojení pak jsou silnice/železnice (pro města), případně datová spojení (pro počítače v síti). V základní verzi budeme předpokládat, že jedno spojení propojuje vždy dvojici míst (např. vlak, který jezdí pouze z konečné na konečnou). V bonusovém testu pak jedno spojení může propojovat i více míst (například vlak zastavující ve více městech). Naše třída `CAccess` si bude pamatovat takovou mapu. Protože o uzlech ani spojeních nebude nic předpokládat, bude třída realizovaná jako šablona a typ uzlu a typ spojení budou generické parametry této třídy.

Ze zadané mapy bude třída umět vypočítat seznam všech míst, kterých lze dosáhnout pomocí dříve zadaných spojení. Výpočet bude navíc možné rozšířit tím, že půjde omezit maximální počet použitých spojení (například výpočet omezíme na max. 5 spojení - tj. max. 4 "přestupy") a dále půjde při výpočtu omezit, která spojení budou uvažovaná (filtr na spojení) - např. omezíme výpočet na vlaky s vyšší než zadanou rychlostí.

Třída bude mít metodu pro přidávání spojení a metodu pro vyhledávání. Kompletní rozhraní `CAccess` bude obsahovat:

implicitní konstruktor
: vytvoří prázdnou instanci `CAccess`,

metoda `Add(e, u1, u2)`
: metoda přidá spojení do `CAccess`. Spojení spojuje dvojici uzlů `u1` a `u2`. Parametr `e` udává parametry spojení. Přidávané spojení je obousměrné.

metoda `Find ( u[, max [, filter]] )`

: Metoda nalezne všechna ostatní místa (uzly) dostupné ze zadaného místa `u`. Pokud nebude přidaný žádný parametr, bude při vyhledávání uvažovat všechna zadaná spojení. Pokud je přidaný parametr `max`, bude uvažovat nejvýše `max` různých spojení (např. pro `max = 1` bude uvažovat pouze místa přímo dostupná nějakým spojením z `u`, pro `max = 2` bude uvažovat pouze místa dostupná kombinací nejvýše dvou navazujících spojení, tedy nejvýše jeden "přestup", pro `max = 3` bude uvažovat pouze místa dostupná kombinací nejvýše tří navazujících spojení, tedy nejvýše dva "přestupy", ...). Pokud bude zadaný i třetí parametr - `filter`, budou do hledání zahrnuté pouze ta spojení, která vyhoví tomuto zadanému filtru, ostatní spojení nebudou uvažována. Filtr bude zadaný v podobě ukazatele na filtrovací funkci, funktoru, nebo lambda výrazu. Filtr při vyvolání pro zadané spojení vrátí hodnotu `true` nebo `false`, podle této hodnoty se spojení buď bude nebo nebude uvažovat.

: Výsledkem volání bude mapa, kde klíče jsou všechna dostupná místa a hodnotou je celé číslo udávající nejmenší počet spojení potřebný pro jeho dosažení z `u`. Pokud je zadané místo `u` neznámé (nebylo metodou `Add` zadané dříve v žádném spojení), bude výsledkem volání metody vyhození výjimky `invalid_argument` (deklarace této výjimky je ve standardní knihovně).

metoda `Add(e, u1, u2, u3, ...)`
: tato metoda je testovaná pouze v bonusovém testu. Metoda je rozšířením povinné metody `Add`, jejím zavoláním se do `CAccess` přidá spojení, které propojuje více míst - `u1`, `u2`, `u3`, ... Zadání lze chápat i tak, že pro účely výpočtu je každá dvojice takto zadaných míst propojena přímo.

Generický parametr `_T` popisuje místa. Pro místa máte garantované následující operace:

- kopírující konstruktor,
- operátor =
- relační operátory (==, !=, < <=, > a >=),
- destruktor,
- operátor pro výpis (<<).
- Další operace s datovým typem nejsou zaručené (mohou, ale nemusí existovat).

Generický parametr `_E` popisuje spojení. Pro spojení máte garantované následující operace:

- kopírující konstruktor,
- operátor =
- destruktor.
- Další operace s datovým typem nejsou zaručené (mohou, ale nemusí existovat).

Odevzdávejte zdrojový kód s implementací šablony třídy `CAccess`. Za základ implementace použijte přiložený zdrojový kód. Pokud v kódu ponecháte bloky podmíněného překladu, lze takový zdrojový kód lokálně testovat a zároveň jej odevzdávat Progtestu.

Hodnocení je rozděleno na povinnou, nepovinnou a bonusovou část. V povinné části se testují mapy s malým množstvím míst. Nepovinný test testuje velké mapy, které vyžadují použití odpovídajících datových struktur STL. Nezvládnutí nepovinného testu znamená citelnou bodovou ztrátu.

Pro hledání cesty se nejlépe hodí algoritmus BFS (prohledávání do šířky). Pro inspiraci si připomeňte proseminář PA1, ve kterém jsme probírali datové struktury, příklad s cestou v bludišti.

Zvládnutí bonusového testu předpokládá využití variadických šablon z C++ 11. Pokud se rozhodnete bonusový test přeskočit, ponechte vypnutou direktivu preprocesoru `MULTIPLE_STOPS` (bude-li direktiva zapnutá bez odpovídající implementace, program nepůjde přeložit). Naopak, pokud se rozhodnete bonusový test řešit, direktivu zapněte (bez jejího zapnutí bude bonusový test přeskočen a bude vždy hodnocen 0 body).

Zdrojový kód s ukázkou práce šablony naleznete v přiloženém archivu.