NPRG041 – 2023/24 ZS – Cvičení MS – **Malý úkol C01**

Podmnožiny

Cílem prvního domácího úkolu je naprogramovat jednoduchou aplikaci, která pro zadanou množinu prvků najde a vypíše všechny její podmnožiny. Abychom si co nejvíce zjednodušili situaci, budeme předpokládat, že prvky této množiny jsou jen písmena anglické abecedy. Samotnou vstupní množinu pak zadáme v podobě obyčejné lokální proměnné přímo v těle funkce `main`, a to formou klasického céčkového pole `const char items[] = { 'A', 'B', 'C', 'D' }`.

Je zřejmé, že budeme muset vhodně pracovat i s velikostí tohoto vstupního pole. A jelikož chceme vytvořit univerzálně použitelný a dobře strukturovaný kód, určitě ji nikde nechceme napevno zadrátovat. Pokud bychom totiž počet prvků zadali jako fixní číslo, museli bychom pak takový údaj manuálně měnit při každé změně vstupní množiny. Použijeme proto následující trik: `const size_t count = sizeof(items) / sizeof(items[0])`.

Celý program dekomponujeme do vhodně navržených funkcí a zcela se vyvarujeme použití globálních proměnných. Všechna potřebná data tedy budeme při volání funkcí předávat pomocí parametrů očekávaných jejich rozhraními. Hlavní funkce odpovídající za celý proces hledání a vypisování podmnožin musí být navržena univerzálně. Budeme ji tedy moci potenciálně volat opakovaně, a to i pro jakékoli jiné nebo jinak velké vstupní množiny. V odevzdaném kódu ale budeme předpokládat vstup jediný, a to ten výše uvedený.

Abychom rozumně pracovali s pamětí a současně zajistili, že budeme schopni všechny možné podmnožiny najít ve stejném pořadí, budeme je generovat pomocí prohledávání do hloubky. Konkrétně tak, že u každého prvku v pořadí zleva doprava rozhodneme, jestli jej v aktuální podmnožině chceme, nebo naopak nikoli. Pořadí prvků nesmíme měnit, přítomnost daného prvku pak má přednost před jeho vynecháním.

Příznaky u jednotlivých prvků si budeme pamatovat pomocí pole logických hodnot. Jeho velikost ale nemůžeme předem znát, takže takové pole budeme muset vytvořit pomocí mechanismu dynamické alokace. Konkrétně stačí použít konstrukci `bool* signature = new bool[count]`, čímž získáme ukazatel na první pozici takového pole. Na samotném konci ale nesmíme zapomenout toto pole zase dealokovat, a to pomocí `delete[] signature`. Pokud bychom to neudělali, danou paměť bychom nevratně ztratili.

Nalezené podmnožiny budeme postupně vypisovat na standardní výstup, a to v následujícím formátu: `{ A, C, D }`. Celou množinu tedy ohraničíme pomocí páru složených závorek, jednotlivé prvky budeme oddělovat čárkou. Pozor musíme dát na mezery, za posledním prvkem už žádnou čárku nepíšeme a v případě prázdné množiny vypíšeme jen mezeru jednu `{ }`. Na každý řádek umístíme právě jednu podmnožinu, každý řádek ukončíme pomocí `std::endl`.

Řešení vypracujte do jednoho jediného `*.cpp` souboru, ten také jako jediný soubor odevzdáte. Řešení musí být správné, předdefinované testy musí skončit úspěšně bez jakékoli chyby. Také je potřeba zařídit, že během kompilace nevzniknou žádná varování, natož chyby. Cílem úkolu je ověřit především schopnost práce s uživatelsky definovanými funkcemi, předáváním argumentů hodnotou nebo ukazatelem, klasickými poli a standardním výstupem. Je potřeba se zaměřit nejenom na věcnou správnost, ale také na kvalitu kódu, zejména vhodnou dekompozici do funkcí nebo jejich rozhraní.

Na závěr se ještě podíváme na několik konkrétních požadavků plynoucích z častých chyb nebo nevhodného návrhu. Konkrétně nepoužívejte pokročilejší konstrukty, se kterými jsme na cvičení ještě nepracovali. Jinými slovy si plně vystačíme s knihovnou `<iostream>`, operátorem `<<` pro zápis do streamu standardního výstupu `std::cout` a klasickým céčkovým polem. Jiné knihovny nepoužívejte, zejména tedy žádné kontejnery apod. Nepoužívejte ani třídy nebo šablony, cílem je vyřešit náš problém adekvátními prostředky.

Nepoužívejte více než jedno dynamicky alokované pole pro naší signaturu. Každou konkrétní podmnožinu vygenerujte právě jen jednou, není přípustné generovat případné duplicity s jejich následným dohledáváním a odstraňováním. Vypsání prvků konkrétní podmnožiny musíte zvládnout jen na jeden průchod.

Názvy všech funkcí a proměnných volte jako dostatečně vysvětlující, vše včetně komentářů pište výhradně v angličtině. U každého parametru funkce zvažte, jestli by neměl být opatřen příznakem `const`. Pro nejrůznější počty nebo velikosti vždy volte typ `size_t`.

Také se ujistěte, že každá funkce plní jen jeden dobře ohraničený a definovaný úkol, tedy že jste minimálně oddělili funkci na hledání podmnožin od jejich vypisování. Současně chcete koncovým uživatelům nabídnout funkci s přívětivým rozhraním, takže takovou veřejnou funkci musíme oddělit od interní rekurzivní. Mimo jiné i proto, že nechceme nikoho nutit alokovat a dealokovat naše čistě interní pole signatury, natož o něm vůbec vědět.