Név: Dombóvári Bálint, Neptun kód: E0PKKH

Specifikáció,Terv:

A program egy ruházati boltot reprezentál. A vásárló a program segítségével lekérdezheti, hogy milyen termékek vannak a boltban, melyekből bevásárló listát készíthet, és végül vásárolhat is. A vásárlás előtt a főprogramban feltöltődik az árukészlet (dinamikus adattárolás), majd (heterogén kollekció formájában) a bevásárló listán gyűlnek össze a vásárló által kiválasztott termékek, végül a vásárlással a raktár készlete is változik. Ha a vásárló meggondolja magát, akkor a bevásárló lista törlődik.

Fájlok:

Shipment.txt: Input fájl, amelyben a raktárba beérkezett termékek vannak. A main.cpp-ben található *bevetelezes()* nevű függvény ebből a fájlból olvassa be az adatokat a fájl végéig, és tölti fel a paraméter listán kapott *Store* osztály példányának *stock[]* tömbjét dinamikus adattárolással. A függvény ellenőrzi, hogy sikerült-e a paraméterlistán kapott fájlt megnyitni (a fájl nevét kapja a függvény). A fájl felépítésében lényeges, hogy az első sorban egy betű található, ami jelöli, hogy a következő termék egy ruha (C), egy cipő (S), vagy egy kiegészítő (A). Mivel a program nem tudja előre, hogy a következő termék milyen típusú, így nem szerencsés, de *switch*-re van szűkség, majd ezután beolvassa a következő sorban lévő adatokat, melyek sorrendje a következő:

1. Név: szóköz helyett egy pont választja el a több szóból álló neveket (*string*)
2. Az adott termék ára (*int*)
3. Mennyiség (*int*)
4. A kedvezmény az adott termékre (*int*)
5. Mérete az adott terméknek (*char*/*int*/*ONE_SIZE*)

Részlet a fájlból:

...

A
NIKE.BAG 20000 2 80
C
ADIDAS.POLO 10000 1 100 M
C
NIKE.POLO 10000 1 100 XS
S
JORDAN 40000 3 100 45
S
AIR.MAX 50000 3 100 43.5
A
PUMA.CAP 5000 3 50
...

Receipt.txt: Output fájl, melybe a vásárlás eredménye fog szerepelni. A *Store* osztály *endShopping()* függvénye paraméter listán kapja meg a *stream*-et, aminek segítségével összegzi a függvény a vásárlást akár a standard output-ra, vagy egy fájlba (*Receipt.txt*).

Osztályok:

(Az eltérések, amelyek a korábbi dokumentációkban voltak, kék színnel vannak kiemelve!)

Product:

Absztrak alaposztály, ami az öröklési hierarchia kialakításában vesz részt. Az összes attribútuma privát (*price*: int, *sale*: int, *quantity*: int, *name*: string), és számos *get*-er függvénnyel is rendelkezik (*getPrice, getSale, getQ, getName, virtual getSizeI, virtual getSizeC*). *DecreaseQ* függvénye egy *set*-er függvény a *quantity* változó értékének módosítására, egy int-et kap, és annyival csökkenti a *quantity* értékét. Vásárlás esetén hívódik meg ez a függvény a *Store* osztály *endShopping()* vagy *pick()* függvényében.

Operatorok közül az *operator*== - vel rendelkezik az osztály, melyben a *price*, *sale*, *quantity* változók alapján dől el az egyenlőség két *Product* példány között,

4 virtuális függvénnyel rendelkezik az osztály (*printSize(), getSizeI(), getSizeC(), operator*==), melyekből csak a *printSize()* tisztán virtuális, és a leszármazott osztályokban van definiálva. Ezek a függvények így az alaposztály felől is elérhetőek.

Konstruktor-a 3 int-et, és egy *string*-et kap, és default értékei is vannak a paraméterlistáján, a *destruktor* pedig virtuális.

A *Product.h* fájlban egy globális *inserter* függvény is található, mely egy *Product* példány nevét, árát és leértékelését írja ki egy *stream*-re. A mennyiséget nem minden függvényben kell kiírni a *Store* osztály függvényeiben, így azt külön a *getQ()* függvény segítségével írja ki a program ha szükség van rá.

Korábban: Másoló *konstruktor* a és *operator*= fv.-e is van az osztálynak.

Cloth, Shoe, Accessory:

Mindegyik leszármazott osztálynak van egy *size* változója, melynek típusa osztályonként eltér.

Az összes osztályban definiálva van a *printSize()* függvény, mely kiírja az adott objektum *size* változóját a paraméterlistán kapott *stream*-re.

Korábban: Másoló *konstruktor* a és *operator*= fv.-e is van az osztályoknak. Az *Accessory* osztályban a *size* típusa const char*

<u>Cloth osztály:</u>

- *char size*: méret
- *int X*: a méretben lévő X-ek száma, default 0
- *Konstruktor*-a 4 *int*-et, 1 *char*-t és egy *string*-et kap, és meghívja az alaposztály *konstruktor*-át.

– A *getSizeC()* függvénye a mérete adja vissza (*char*), míg *getSizeI()* függvénye a X-ek számát adja vissza a méretben.

- *Operator*== fv.-e a termékek méretet is ellenőrzi.
- *Deskturktor*-ból az alapértelmezett is jó

Shoe osztály:

- *double size*: fél méretek is lehetségesek
- *Konstruktor*-a 3 *int*-et , 1 *double*-t és egy *string*-et kap, meghívja az alaposztály *konstruktor*-át.
- Csak a *getSizeI()* függvény van felüldefiniálva, mely a méretet adja vissza.
- *Operator*== fv.-e a termékek méretet is ellenőrzi.
- *Deskturktor*-ból az alapértelmezett is jó

Accessory osztály:

- *static string size*: mivel minden kiegészítő méret ONE_SIZE, így praktikus a *static*
- *Konstruktor*-a 3 *int*-et, és egy *string*-et kap, meghívja az alaposztály *konstruktor*-át.
- *Operator*== fv-ét nem kell felül definiálni, mert nem kell a méreteket ellenőrizni, így jó *Product* osztályban lévő virtuális verzió is.
- *Deskturktor*-ból az alapértelmezett is jó

Store:

Heterogén kollekciót megvalósító osztály, melyben a tömbök mérete *template*-en keresztül szabályozható. *size_t MAX* a *stock[]* tömb méretét adja, míg *size_t max* a *list[]* méretére vonatkozik, és mind a kettő tömb *Product* pointerekt tárol. Default értékek: *MAX* = 100, *max* = 20. Másoló *konstruktor*-a és *operator*= függvénye privát, mondván az alapértelmezett nem jó. *Konstruktor*-ában 0 értéket kap a *listdb*, és a *stockdb* nevű változója,

melyek azt jelölik, hogy az névnek megfelelő tömbben éppen mennyi elem található aktuálisan. A *stock[]* tömb feltöltése a *main*-ben fog történni egy fájlból történő beolvasással a *bevetelezes()* függvény segítségével, ez lesz a raktárkészlet, míg a *list[]* tömb a bevásárló listát valósítja meg.

makeStock (void): A függvény paraméterlistán az új elemre mutató pointert veszi át, melynek dinamikusan foglaltunk helyet a fájlból történő beolvasás során. Ellenőrzi, hogy van e elegendő hely a raktárba az új terméknek, és ha igen, akkor hozzá adja a raktárhoz. Ha nincs elegendő hely akkor kivételt dob, miután felszabadított a dinamikusan foglelt területet.
available (bool): egy adott termék elérhetőségét lekérdező függvény, mely paraméterként egy terméket (*const Product&*) és egy *stream*-et (*ostream&*) kap, majd ellenőrizi, hogy elérhető-e az adott termék a raktárban, és ennek függvényében tér vissza egy bool értékkel. Ha elérhető, akkor kiírja a nevét árát, méretét, leértékelését, és az elérhető mennyiséget a vásárló számára a kapott *stream*-re, ha pedig nincs a raktárban, akkor erről is tájékoztatást ad.
pick (void): a kiválasztott terméket a bevásárló listához csatoló függvény. Egy terméket (*const Product&*), és egy mennyiséget kap (*int*) a paraméterlistán, és először ellenőrzi, hogy van e elegendő hely a kosárban a terméknek. Ha nincs, vagy a megadott mennyiség irreális akkor kivételt dob, egyébként megkeresi a kívánt terméket és csak akkor adja hozzá a kosárhoz, ha van legalább annyi belőle, mint amennyit kértek. Dinamikusan nem foglal újabb területet a függvény, hanem tegyük fel, hogy az adott termék a *stock[]* tömbben az i-dik helyen van, akkor *list[]* tömb következő pointere egyenlő lesz a *stock[i]* helyen lévő pointerrel. Végül a függvény tájékoztatást ad a folyamat eredményéről a paraméterlistán kapott *stream*-en keresztül.
checkOut (void): tájékoztatja a vásárlót az eddig kiválasztott termékekről. A termékek nevét, méretét, kedvezményét, árát, és a végén az összegzett árat is kiírja a paraméter listán kapott *stream*-re (*ostream&*).

endShopping (void): a rendelést rögzítő/elutasító függvény. Egy *bool* értéket kap, ami a vásárlásról hozott döntést jelenti, és egy *stream*-et (*ostream&*), amire kiírja az összegzést. Vásárlás esetén kiüríti a kosarat, azaz a *list[]* tömb pointereit NULL-ra állítja. A vásárlás elutasítása esetén pedig nem csak kiüríti a kosarat, hanem a termékek “visszakerülnek a polcokra” úgy, hogy a *Product* osztályban lévő *decreaseQ* függvényt (-1) értékkel hívjuk meg.

getListdb (size_t), getStockdb (size_t): *get*-er függvények az aktuális készlet és kosár tartalmának lekérdezéséhez.

getListMax (size_t), getStockMax (size_t): Az adott raktárra vonatkozó maximális kapacitást, és a bevásárló lista maximális méretét adja vissza ez a két függvény. Szükség van rájuk, mert ezek az értékek (*MAX,max*) raktáronként eltérhet.

printStock (void): A paraméterlistán kapott *stream*-re (*ostream&*) kiírja a raktárkészletet.

Destruktor: a dinamikusan foglalt területek felszabadítása

Korábban:

- A *static* adattagok korlátozzák a tömbök méretét: *stock[100], list[20]*.
- Egy példány fog belőle generálódni a főprogram elején.
- *Pick(void)*: a *list[]* tömbhöz csatolást dinamikus memóriakezeléssel valósítja meg.
- *makeStock (void)*: Ez a függvény hívódik meg a *main* elején és dinamikus módon tölti fel a tömböt egy txt fájlból. Paraméterlistán veszi át a fájl nevét.
- *endShopping*: adott esetben dinamikus memória kezelést végez (pl.: ha egy adott termék elfogy). A vásárlás végeredményét kiírja egy fájlba.
- *operator[]* is rendelkezik az osztály.

Főprogram:

A főprogram két részből áll. Az első részben a program saját maga teszteli az osztályok függvényeit a *Store* osztály kivételével különböző teszteseteken keresztül, melyek kihasználják a *gtest_lite.h* fájl által nyújtott lehetőségeket:

1. Teszt: default értékkel létrehozott *const Accessory* példány *getPrice(), getSale(), getQ(), getName()* függvényének tesztje
2. Teszt: részben default értékkel létrehozott *Shoe* példány *getPrice(), getSale(), getQ(), getName()* függvényének tesztje
3. Teszt: Nem default értékekkel létrehozott *Cloth* példány *getPrice(), getSale(), getQ(), getName()* függvényének tesztje
4. Teszt: A korábban létrehozott objektumok *printSize()* függvényének tesztje
5. Teszt: Mutatókonverzió során a *getPrice(), getSale(), getQ(), getName()* és *printSize()* függvény tesztje

A második részben következik a Store osztály tesztje, melynek kettő verziója van.

Az első verzióban még a gép maga teszteli az osztály függvényeit egy példányon: *getListdb(), getStockdb(), getListMax(), getStockMax(), makeStock(), available(), pick(), endShopping(), printStock()*. Ebben részben a *main.cpp*-ben található *bevetlezes()* függvény sikerességét is nézi a program, amely a *Shipment.txt* fájlból olvas, és a *Store* osztály *makeStock()* függvényén keresztül tölti fel a paraméterlistán kapott *Store* példány raktárkészletét. Végül két tesztesetben a kivételkezelést is ellenőrzi a *main*.

A második verzió egy interaktív teszt, melyhez egy felhasználóra van szükség, így a JPortára feltöltött verzióban ez a funkció nem elérhető, csak a személyes bemutatáson .

Ez a program rész csak a korábban említett függvényeket használja a *hasznalat()* függvény kivételével, mely kiírja standard output a felhasználói teszt használati útmutatóját:

1.gomb: Adott termék elérhetőségének lekérdezése.

2.gomb: Adott termék hozzáadása a kosárhoz.

3.gomb: Check-out.

4.gomb: Vásárlás megerősítése/elutasítása.

5.gomb: Az árukészlet lekérdezése a helyes működés ellenőrzéséhez

9.gomb: Programleállítása.

A programról elmondható általánosan, hogy magyar nyelven működik, az árak forintban vannak megadva és a méretezés európai.