# بقالی

در کد نوشته شده برای این سوال دو کلاس و یک تابع "main"تعریف شده است

## کلاس رابط کاربری

1. class UI

2. {

3. public:

4. void terminal(HSD &hsd)

5. {

6. string cmd = "", arg0 = "";

7. int arg1[2] = {0, 0};

8. while (true)

9. {

10. cmd = "";

11. arg0 = "";

12. arg1[0] = 0;

13. arg1[1] = 0;

14. cout << "$ ";

15. cin >> cmd;

16. if (cmd == "Define")

17. {

18. cin >> arg0 >> arg1[0] >> arg1[1];

19. hsd.Define(arg0, arg1[0], arg1[1]);

20. }

21. else if (cmd == "Delete")

22. {

23. cin >> arg0;

24. hsd.Delete(arg0);

25. }

26. else if (cmd == "Sell")

27. {

28. cin >> arg0 >> arg1[0];

29. hsd.SellF(arg0, arg1[0]);

30. }

31. else if (cmd == "Buy")

32. {

33. cin >> arg0 >> arg1[0];

34. hsd.BuyF(arg0, arg1[0]);

35. }

36. else if (cmd == "Status")

37. {

38. hsd.Status();

39. }

40. else if (cmd == "Financial")

41. {

42. hsd.Financial();

43. }

44. else if (cmd == "Exit")

45. {

46. return;

47. }

48. else

49. {

50. cout << "Command Not Found !\n";

51. }

52. }

53. }

54. };

کد کلاس رابط کاربری

در این کلاس فقط یک تابع ترمینال (terminal) تعریف شده است که کار این تابع درست کردن یک محیط شبیه سازی شده از ترمینال دستور خطی هست

عملکرد این تابع به این صورت است که یک حلقه while که شرط همیشه صحیح برای آن گذاشته شده است قرارداده شده تا موقعی که تابع Exitخوانده نشود همیشه در حال چرخش هست. در تابع سه متغییر cmd که دستور داده شده در آن ذخیره میشود و arg0 که یک متغیر از نوع رشته برای ذخیره ورودی نام برای هر دستور و ارایه arg1 که از نوع integer هست و برای ذخیره اعداد ورودی (به‌طور مثال مبلغ خرید و فروش کالا) استفاده شده و باتوجه به دستور داده شده ورودی های لازم به توابع لازمه در کلاس حسابداری (hsd) می‌فرستد و برای هر دور استفاده علامت $ را چاپ میکند.

## کلاس کالاها

1. class Stuff

2. {

3. public:

4. int BC;

5. int SC;

6. int Rep;

7. string Name;

8. Stuff(int bc,int sc,int rep,string name)

9. {

10. BC = bc;

11. SC = sc;

12. Rep = rep;

13. Name = name;

14. }

کلاس تعریف متغییر های یک کالا

BC & SC : لیست مبالغ خرید و فروش کالاها

Rep : لیست موجودی انبار کالاها

Name : لیست نام کالاها

## کلاس حسابداری

### متغیر های تعریف شده

1. class HSD

2. {

3. private:

4. vector<Stuff> stuff;

5. long Buy = 0, Sell = 0;

متغییر های کلاس

کلاس کالا را به صورت یک لیست در متغییر های تعریف شده ، تعریف کردم.

Buy & Sell :‌ مبالغ خرید و فروش تا هر لحظه ی استفاده شده از نرم افزار (long گرفتم چون احتمال بالا رفتن اعداد رو دادم)

و عملا جایگاه هر کالا در لیست همآنند ID هر سطر در جدول SQL هست.

بنابراین داریم که اگر جایگاه نام کالا را پیدا کنیم قیمت ها و موجودی آن را پیدا کردیم

پس تابع پیدا کردن جایگاه کالا را بر حسب نام آن را نوشتم.

## پیدا کردن جایگاه کالا

1. int Find(string nam)

2. {

3. for (int i = 0; i < stuff.size(); i++)

4. {

5. if (stuff.at(i).Name == nam)

6. return i;

7. }

8. return -1;

9. }

تابع پیدا کردن جایگاه کالا در لیست

این تابع به جستوجو جایگاه کالا گشته و اگر آن کالا در لیست وجود نداشته باشد جواب پرت ۱- را خارج می‌کند.

## تعریف کردن کالا

1. void Define(string nam, int buy, int sel)

2. {

3. if (Find(nam) != -1)

4. {

5. cout << "!\n";

6. return;

7. }

8. Stuff tmp(buy, sel, 0, nam);

9. stuff.push\_back(tmp);

10. return;

11. }

تابع تعریف کردن کالا

همان طور که دیده می‌شود یک شرط ، اول این تابع گذاشته شده است که در صورت وجود آن کالا ، اجازه باز تعریفی آن کالا را به کاربر بدهد و در ادامه مقادیر لازم به لیست ها اضافه گردند.

## حذف کالا

1. void Delete(string nam)

2. {

3. int loc = Find(nam);

4. if (loc == -1)

5. {

6. cout << "!\n";

7. return;

8. }

9. if (stuff.at(loc).Rep != 0)

10. {

11. cout << "!\n";

12. return;

13. }

14. stuff.erase(stuff.begin() + loc);

15. }

16.

تابع حذف کالا

اول این تابع چک می‌کند که آن کالا در لیست وجود دارد یا نه و بعد از آن اگر وجود داشت چک می‌کند که کالا موجودی نداشته باشد و اگر موجودی هم نداشت آن کالا را از لیست پاک می‌کند.

## فروش کالا

1. void SellF(string nam, int count)

2. {

3. int loc = Find(nam);

4. if (stuff.at(loc).Rep <= count)

5. {

6. cout << "!\n";

7. return;

8. }

9. stuff.at(loc).Rep -= count;

10. Sell += stuff.at(loc).SC \* count;

11. return;

12. }

تابع فروش کالا

این تابع اول چک می‌کند که به اندازه لازم ، کالا در انبار برای فروش وجود داشته باشد ، درصورت موجودی ، مقدار لازم آن کالا را از انبار کم و حساب فروش را به اندازه مبلغ فروش بالا میبرد.

## خرید کالا از کارخانه

1. void BuyF(string nam, int count)

2. {

3. int loc = Find(nam);

4. if (loc == -1)

5. {

6. cout << "!\n";

7. return;

8. }

9. stuff.at(loc).Rep += count;

10. Buy += stuff.at(loc).BC \* count;

11. }

تابع خرید کالا از کارخانه

اول این تابع چک می‌کند که آن کالا در لیست وجود دارد یا نه و بعد از آن اگر وجود داشت مقدار لازم را از حساب خرید کالا کم و به انبار اجناس را اضافه می‌کند.

## وضعیت کالا ها

1. void Status()

2. {

3. for (int i = 0; i < stuff.size(); i++)

4. {

5. cout << stuff.at(i).Name << " " << stuff.at(i).BC << " " << stuff.at(i).SC << " " << stuff.at(i)Rep << endl;

6. }

7. return;

8. }

تابع وضعیت کالا ها

این تابع همان طور که معلوم است به ازای هر کالا در لیست وضعیتش را چاپ میکند.

## صورت مالی

1. void Financial()

2. {

3. cout << Sell << " " << Buy << " " << Sell - Buy << endl;

4. }

تابع صورت مالی

این تابع همان طور که معلوم است میزان خرید، میزان فروش و سود را چاپ می‌کند.

## تابع main

1. int main()

2. {

3. HSD hsd;

4. UI ui;

5. ui.terminal(hsd);

6. return 0;

7. }

تابع main

در این تابع تعریف کلاس های مورد نظر و فراخوانی رابط کاربری انجام شده است.