C++프로그래밍및실습

편의점 매대 관리 시스템

중간보고서

제출일자: 2024.11.17.
제출자명: 김태균
제출자학번: 213961

1. 프로그램 실행방법

A. storage.csv파일과 main.exe파일을 같은 폴더에 두고 exe폴더를 cmd로 실행

2. 기능별 구현상황

A. 재고 관리 기능

i. 데이터베이스 읽기

ii. 재고 추가

iii. 상품 추가

iv. 상품 제거

v. 관리 명령어 입력창

vi. 저장 및 종료 기능

vii. 데이터베이스 출력

B. 매대 상품 수량 변경 기능

i. 상품 판매

ii. 매대 진열

C. 진열 필요 상품 제시 기능

i. 매대 상태 및 피크 타임 여부 판단, 진열 필요 상품 제시

3. 설계 및 구현

A. 재고 관리 기능 – 데이터베이스 읽기

i. 코드 스크린샷

storage.h

```
9   private:
10      fstream fs;
11      string storage_name;
12      vector<vector<string>> database; // 2차원 벡터를 활용한 동적 배열
13  public:
14      Storage(string filename);
15      Storage(const Storage& other); // database 멤버 변수를 깊은 복사하기 위한 복사생성자
16      ~Storage(); // 소멸시 변경된 내용을 저장하는 소멸자
17
18      vector<string> getItem(string code) const; // code에 해당하는 상품을 vector로 반환
19      vector<string> getItem(int line) const; // line번째 상품을 벡터로 반환
20      int size() const; // database의 크기 반환, 목록 부분을 제외한 사이즈이다
21
22      void addLine(vector<string> new_line); // database 행 추가
23      void removeItem(string code); // code에 해당하는 제품 삭제
24      void itemSold(string code, int qnt); // code에 해당하는 제품 판매
25      void itemToStand(string code, int qnt); // code에 해당하는 제품 qnt만큼 매대에 진열
26      void itemStore(string code, int qnt); // code에 해당하는 제품 qnt만큼 재고수 추가
27      void printDatabase(); // database 출력
28  };
```

storage.cpp

```
Storage::Storage(string filename)
{
    string str_buf;
    this->storage_name = filename;
    this->fs.open(filename, ios::in);
    while(!fs.eof())
    {
        vector<string> tmp;
        getline(this->fs, str_buf,'\n');

        int position;
        int cur_position = 0;
        while((position = str_buf.find(",", cur_position)) != string::npos)
        {
            int len = position - cur_position;
            tmp.push_back(str_buf.substr(cur_position, len));
            cur_position = position + 1;
        }
        tmp.push_back(str_buf.substr(cur_position));
        addLine(tmp);
    }
}
```

```cpp
Storage::Storage(const Storage &other)
{
    this->fs.open(this->storage_name);
    this->storage_name = other.storage_name;
    for(vector<string> v : other.database)
    {
        vector<string> tmp;
        for(string data : v)
        {
            tmp.push_back(data);
        }
        addLine(tmp);
    }
}
```

```cpp
vector<string> Storage::getItem(string code) const
{
    for(vector<string> v : this->database)
    {
        if(v[0] == code)
        {
            vector<string> result;
            for(string data : v)
            {
                result.push_back(data);
            }
            return result;
        }
    }
    cout << "Wrong item code" << endl;
}

vector<string> Storage::getItem(int line) const
{
    vector<string> tmp;
    for(string data : this->database[line + 1]) // 목록부분을 제외하기 때문에 인덱스 값에 +1 을한다
    {
        tmp.push_back(data);
    }
    return tmp;
}

int Storage::size() const
{
    return this->database.size() - 1; // database의 목록부분을 제외한 사이즈
}
```

main.cpp

```cpp
int main(int argc, char** argv)
{
    Storage storage("storage.csv");
```

ii. 입력

- string filename: csv파일의 파일명

iii. 결과

- main함수에서 storage 객체 생성

- csv파일을 읽고 2차원 동적 배열에 저장

iv. 설명

- storage 객체 정의

- storage 객체의 정보를 반환하는 get메서드 정의

- getItem의 경우 상품의 code 또는 목록에서의 인덱스 값으로 검색할 수 있도록 오버로드 함

- storage 생성자를 정의해 파일명에 해당하는 csv파일을 읽고 멤버변수인 vector<vector<string>>에 값들을 저장

- 복사 생성자를 정의해 멤버변수 database를 복사할 때 깊은 복사를 할 수 있도록 설계함

v. 수업 연관 내용

- 7주차 function overload

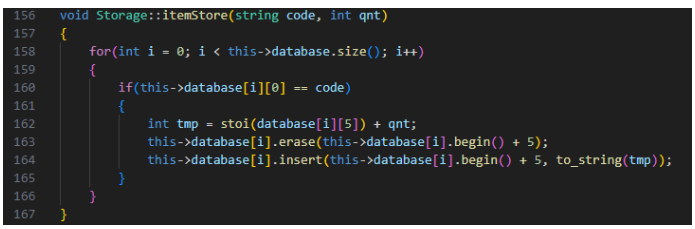
- 10주차 vector

- 11주차 copy constructor

B. 재고 관리 기능 – 재고 추가

i. 코드 스크린샷

storage.cpp



```
156  void Storage::itemStore(string code, int qnt)
157  {
158      for(int i = 0; i < this->database.size(); i++)
159      {
160          if(this->database[i][0] == code)
161          {
162              int tmp = stoi(database[i][5]) + qnt;
163              this->database[i].erase(this->database[i].begin() + 5);
164              this->database[i].insert(this->database[i].begin() + 5, to_string(tmp));
165          }
166      }
167  }
```

ii. 입력

- string code: 재고 추가할 상품의 code

- int qnt: 추가할 재고량

iii. 반환값

- 없음

iv. 결과

- 상품의 재고량을 입력된 개수만큼 추가

v. 설명

- 입력된 code의 상품을 검색한 후 입력된 qnt만큼 재고를 추가해 저장

vi. 수업 연관 내용

- 10주차 vector

C. 재고 관리 기능 – 상품 추가

i. 코드 스크린샷

storage.cpp

```
94  void Storage::addLine(vector<string> new_line)
95  {
96      this->database.push_back(new_line);
97  }
```

ii. 입력

- vector<string> new_line: 새로 추가할 상품의 정보가 담긴 1차원 벡터

iii. 반환값

- 없음

iv. 결과

- 상품을 database에 추가한다

v. 설명

- 입력값인 new_line은 database의 목록 규격에 맞춰서 미리 생성되어 있어야 한다

vi. 수업 연관 내용

- 10주차 vector

D. 재고 관리 기능 – 상품 제거

i. 코드 스크린샷

storage.cpp

```
 99  void Storage::removeItem(string code)
100  {
101      for(int i = 0; i < this->database.size(); i++)
102      {
103          if(this->database[i][0] == code)
104          {
105              this->database.erase(this->database.begin() + i);
106          }
107      }
108  }
```

ii. 입력

- string code: 상품의 code값

iii. 반환값

- 없음

iv. 결과

- code값에 해당하는 상품을 삭제한다

v. 설명

- code값으로 검색하여 해당 코드의 상품을 제거

vi. 수업 연관 내용

- 10주차 vector

E. 재고 관리 기능 – 관리 명령어 입력창

i. 코드 스크린샷

main.cpp

```
12      bool flag = true;
13      int menu;
14      while(flag)
15      {
```

```
61          // 메뉴 선택
62          cout << "1. Sell item  2. Add item  3. Save and Exit" << endl;
63          cin >> menu;
64          switch(menu){
65              case 1:
66              {
67                  string sellItemCode;
68                  int sellQnt;
69                  cout << "Item Code: ";
70                  cin >> sellItemCode;
71                  cout << "Quantity: ";
72                  cin >> sellQnt;
```

```
73                  if(stoi(storage.getItem(sellItemCode)[3]) < sellQnt)
74                  {
75                      cout << "Wrong quantity!";
76                      break;
77                  }
78                  storage.itemSold(sellItemCode, sellQnt);
79                  break;
80              }
81              case 2:
82              {
83                  string itemCode;
84                  string itemName;
85                  string stand;
86                  string curStandQnt;
87                  string maxStandQnt;
88                  string qntStorage;
89
90                  cout << "Item Code: ";
91                  cin >> itemCode;
92                  cout << "Item Name: ";
93                  cin >> itemName;
94                  cout << "Stand that Item belongs: ";
95                  cin >> stand;
96                  cout << "Current Qunatity in Stand: ";
97                  cin >> curStandQnt;
98                  cout << "Max Qunatity of Stand: ";
99                  cin >> maxStandQnt;
100                 cout << "Qunatity in Storage";
101                 cin >> qntStorage;
102
```

```
112             case 3:
113             {
114                 flag = false;
115                 break;
116             }
117         }
```

ii. 입력

- bool flag: 반복문 탈출 flag

- int menu: 명령어 입력 값

iii. 결과

- 입력 받은 메뉴에 따라 상품 판매, 상품 추가, 저장 및 종료 명령을 실행한다

iv. 설명

- itemSold 함수를 호출하여 상품 판매를 처리한다

- addLine 함수를 호출하여 상품 추가를 처리한다

- 저장 및 종료 명령의 경우 flag값을 바꿔 반복문을 탈출, 프로그램이 종료된다. 이때 소멸자에 의해 storage 객체가 소멸하며 변경된 내용을 저장한다

v. 수업 연관 내용

- 4주차 condition

F. 재고 관리 기능 – 저장 및 종료

i. 코드 스크린샷

storage.cpp

```
45  // 소멸하면서 데이터베이스의 내용을 같은 이름의 새로운 csv파일로 만들어 저장한다
46  Storage::~Storage()
47  {
48      ofstream outfile(this->storage_name);
49      for(int i = 0; i < this->database.size(); i++)
50      {
51          for(int j = 0; j < this->database[i].size(); j++)
52          {
53              outfile << this->database[i][j];
54              if(j != this->database[i].size() - 1)
55                  outfile << ",";
56          }
57          if(i != this->database.size() - 1)
58              outfile << endl;
59      }
60  }
```

ii. 입력

- storage_name: 파일명

- database: 객체 멤버 변수인 2차원 동적 배열

iii. 결과

- 변경된 내용으로 동일한 파일명의 파일을 만들어 저장한다

iv. 설명

- 소멸자를 정의하여 main함수가 종료되어 프로그램이 종료되는 시점에 객체가 소멸하며 변경된 내용을 자동으로 저장하게 된다

v. 수업 연관 내용

- 10주차 Deconstructor

G. 재고 관리 기능 – 데이터베이스 출력

i. 코드 스크린샷

storage.cpp

```
169  void Storage::printDatabase()
170  {
171      for(vector<string> v : this->database)
172      {
173          int ctr = 0;
174          for(string s : v)
175          {
176              if(ctr++ != 1)
177                  cout.width(13);
178              else
179                  cout.width(25);
180              cout << std::right << s << " ";
181          }
182          cout << endl;
183      }
184  }
```

ii. 입력

- database: 객체 멤버 변수인 2차원 동적 배열

iii. 반환값

- 없음

iv. 결과

- 데이터베이스 출력

v. 설명

- 형식에 맞춰 출력할 수 있도록 cout.width()로 간격 설정 및 std::right로 간격에 맞춰 출력

H. 매대 상품 수량 변경 기능 – 상품 판매

i. 코드 스크린샷

storage.cpp

```
110 void Storage::itemSold(string code, int qnt)
111 {
112     for(int i = 0; i < this->database.size(); i++)
113     {
114         if(this->database[i][0] == code)
115         {
116             int tmp = stoi(database[i][3]) - qnt;
117             if(tmp < 0)
118             {
119                 cout << "Not enough items on the stand. Check quantity of the sold item" << endl;
120                 return;
121             }
122             this->database[i].erase(this->database[i].begin() + 3);
123             this->database[i].insert(this->database[i].begin() + 3, to_string(tmp));
124         }
125     }
126 }
```

ii. 입력

- string code: 판매할 상품의 code

- int qnt: 판매할 수량

iii. 반환값

- 없음

iv. 결과

- 판매한 수량만큼 매대에서 수량감소

- 매대에 수량이 부족할 경우 잘못된 입력이라고 판단

v. 설명

- 상품의 code로 database를 검색하여 해당 상품의 매대 진열 수량을 변경한다

I. 매대 상품 수량 변경 기능 – 매대 진열

i. 코드 스크린샷

storage.cpp

```
128  void Storage::itemToStand(string code, int qnt)
129  {
130      for(int i = 0; i < this->database.size(); i++)
131      {
132          if(this->database[i][0] == code)
133          {
134              if(qnt > stoi(database[i][5]))
135              {
136                  cout << "Not enough items in the inventory. Check quantity of the item" << endl;
137                  return;
138              }
139
140              int tmp = stoi(database[i][3]) + qnt;
141              if(tmp > stoi(database[i][4]))
142              {
143                  cout << "Not enough space for the item. Check quantity of the item" << endl;
144                  return;
145              }
146              this->database[i].erase(this->database[i].begin() + 3);
147              this->database[i].insert(this->database[i].begin() + 3, to_string(tmp));
148
149              tmp = stoi(database[i][5]) - qnt;
150              this->database[i].erase(this->database[i].begin() + 5);
151              this->database[i].insert(this->database[i].begin() + 5, to_string(tmp));
152          }
153      }
154  }
```

ii. 입력

- string code: 판매할 상품의 code

- int qnt: 매대에 진열할 수량

iii. 반환값

- 없음

iv. 결과

- 재고 상품의 수량이 감소하고 매대에 진열된 수량이 증가한다

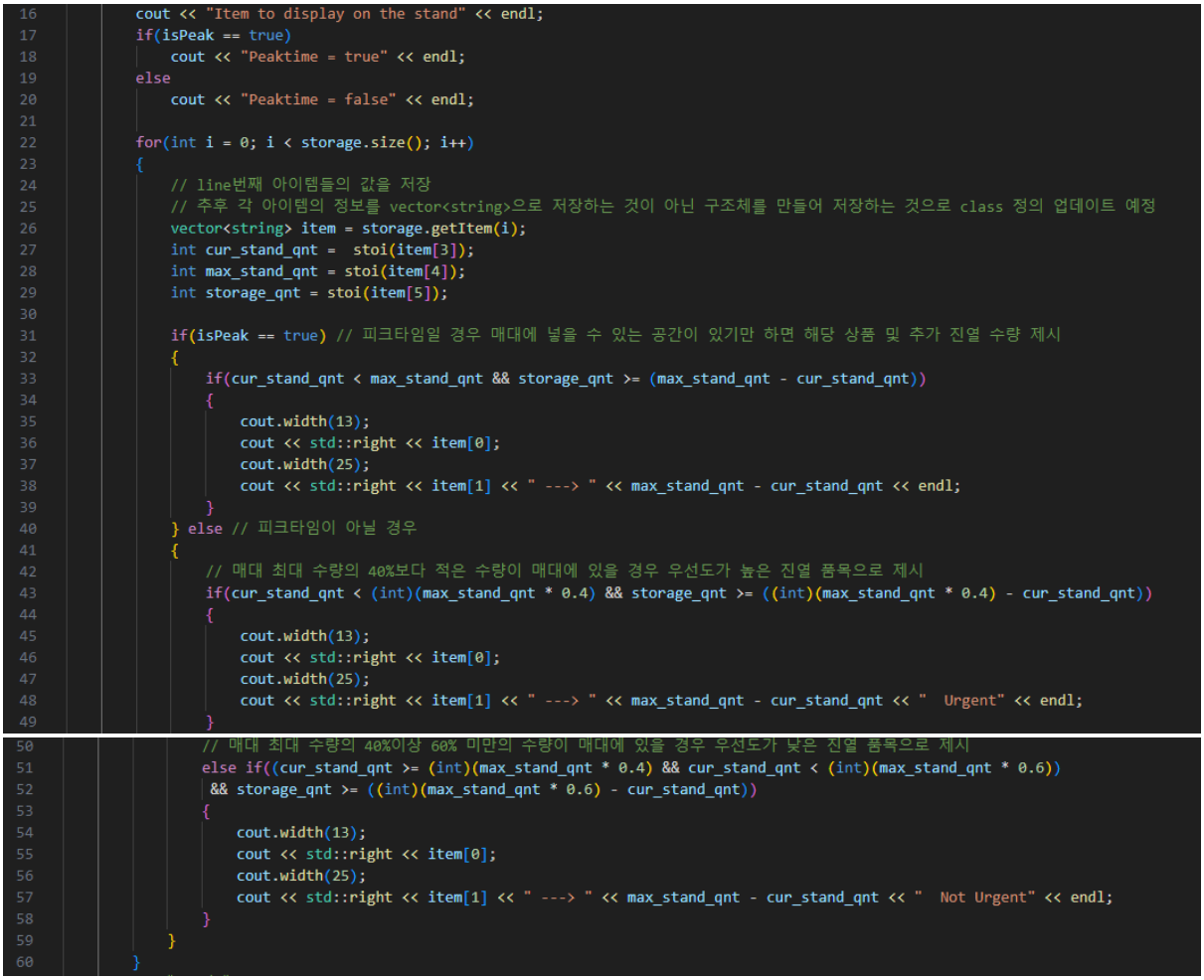
v. 설명

- 재고 수량이 부족하거나 매대에 공간이 부족한 상황에서 너무 많은 수량을 진열하려 한 경우 에러 메시지 출력

J. 매대 상태 및 피크 타임 여부 판단, 진열 필요 상품 제시

i. 코드 스크린샷

main.cpp



```
        cout << "Item to display on the stand" << endl;
        if(isPeak == true)
            cout << "Peaktime = true" << endl;
        else
            cout << "Peaktime = false" << endl;

        for(int i = 0; i < storage.size(); i++)
        {
            // line번째 아이템들의 값을 저장
            // 추후 각 아이템의 정보를 vector<string>으로 저장하는 것이 아닌 구조체를 만들어 저장하는 것으로 class 정의 업데이트 예정
            vector<string> item = storage.getItem(i);
            int cur_stand_qnt =  stoi(item[3]);
            int max_stand_qnt = stoi(item[4]);
            int storage_qnt = stoi(item[5]);

            if(isPeak == true) // 피크타임일 경우 매대에 넣을 수 있는 공간이 있기만 하면 해당 상품 및 추가 진열 수량 제시
            {
                if(cur_stand_qnt < max_stand_qnt && storage_qnt >= (max_stand_qnt - cur_stand_qnt))
                {
                    cout.width(13);
                    cout << std::right << item[0];
                    cout.width(25);
                    cout << std::right << item[1] << " ---> " << max_stand_qnt - cur_stand_qnt << endl;
                }
            } else // 피크타임이 아닐 경우
            {
                // 매대 최대 수량의 40%보다 적은 수량이 매대에 있을 경우 우선도가 높은 진열 품목으로 제시
                if(cur_stand_qnt < (int)(max_stand_qnt * 0.4) && storage_qnt >= ((int)(max_stand_qnt * 0.4) - cur_stand_qnt))
                {
                    cout.width(13);
                    cout << std::right << item[0];
                    cout.width(25);
                    cout << std::right << item[1] << " ---> " << max_stand_qnt - cur_stand_qnt << "  Urgent" << endl;
                }
                // 매대 최대 수량의 40%이상 60% 미만의 수량이 매대에 있을 경우 우선도가 낮은 진열 품목으로 제시
                else if((cur_stand_qnt >= (int)(max_stand_qnt * 0.4) && cur_stand_qnt < (int)(max_stand_qnt * 0.6))
                 && storage_qnt >= ((int)(max_stand_qnt * 0.6) - cur_stand_qnt))
                {
                    cout.width(13);
                    cout << std::right << item[0];
                    cout.width(25);
                    cout << std::right << item[1] << " ---> " << max_stand_qnt - cur_stand_qnt << "  Not Urgent" << endl;
                }
            }
        }
```

ii. 입력

- bool isPeak: 피크타임 여부

- vector<string> item: 상품 정보

- int cur_stand_qnt, max_stand_qnt, storage_qnt: 상품 정보로부터 추출한 현재 매대 진열 수량, 최대 진열 가능 수량, 재고 수

iii. 결과

- 매대 진열된 수량과 최대 진열 가능 주량,

iv. 설명

- 피크타임인 경우: 모든 상품을 최대한 빠르게 진열해야 하기 때문에 우선순위를 제시하지 않고 최대 진열 가능 수량보다 매대 진열 수량이 적을 경우 진열해야 하는 상품과 수량을 제시한다

- 피크타임이 아닌 경우: 매대 최대 수량의 40%보다 적게 진열된 경우 우선순위가 높은 상품, 60%보다 작고 40%이상 진열된 경우 우선순위가 낮은 상품, 두 카테고리로 나누어 진열해야 하는 상품과 수량을 제시한다

- 재고 수 보다 많은 수량을 진열해야 하는 경우는 제시하지 않는다

4. 테스트

A. database 출력

code	name	stand	cur_stand_qnt	max_stand_qnt	inventory
1000000	ShirimpSnack	Snack	1	5	10
1000002	PotatochipOriginal	Snack	3	5	30
1000003	PotatochipOnion	Snack	5	5	7
1000004	CornSnackSpicy	Snack	1	5	0
1000005	CornSnackSweet	Snack	3	5	0
3000000	FishcakebarSpicy	Refrigerated	5	12	0
3000001	FishcakebarBBQ	Refrigerated	4	12	0
3000002	ChickenChest	Refrigerated	6	12	0
4000000	SweetSpicyCupnoodle	Cupnoodle	2	3	12
4000001	SpicyChickenCupnoodle	Cupnoodle	3	3	8
5000000	Sprite	Walkin	7	15	6
5000001	Pepci	Walkin	12	15	0
5000002	PepciZeroLime	Walkin	7	15	0

B. 상품 추가

```
        code                    name         stand cur_stand_qnt max_stand_qnt     inventory
      1000000            ShirimpSnack         Snack             1             5            10
      1000002       PotatochipOriginal         Snack             3             5            30
      1000003          PotatochipOnion         Snack             5             5             7
      1000004           CornSnackSpicy         Snack             1             5             0
      1000005           CornSnackSweet         Snack             3             5             0
      3000000        FishcakebarSpicy  Refrigerated             5            12             0
      3000001          FishcakebarBBQ  Refrigerated             4            12             0
      3000002           ChickenChest  Refrigerated             6            12             0
      4000000      SweetSpicyCupnoodle     Cupnoodle             2             3            12
      4000001    SpicyChickenCupnoodle     Cupnoodle             3             3             8
      5000000                   Sprite        Walkin             9            15             6
      5000001                    Pepci        Walkin            12            15             0
      5000002            PepciZeroLime        Walkin             7            15             0
Item to display on the stand
Peaktime = false
      1000000            ShirimpSnack ---> 4  Urgent
1. Sell item  2. Add item  3. Save and Exit
2
Item Code: 5000003
Item Name: PepciZeroMango
Stand that Item belongs: Walkin
Current Qunatity in Stand: 14
Max Qunatity of Stand: 15
Qunatity in Storage: 20
        code                    name         stand cur_stand_qnt max_stand_qnt     inventory
      1000000            ShirimpSnack         Snack             1             5            10
      1000002       PotatochipOriginal         Snack             3             5            30
      1000003          PotatochipOnion         Snack             5             5             7
      1000004           CornSnackSpicy         Snack             1             5             0
      1000005           CornSnackSweet         Snack             3             5             0
      3000000        FishcakebarSpicy  Refrigerated             5            12             0
      3000001          FishcakebarBBQ  Refrigerated             4            12             0
      3000002           ChickenChest  Refrigerated             6            12             0
      4000000      SweetSpicyCupnoodle     Cupnoodle             2             3            12
      4000001    SpicyChickenCupnoodle     Cupnoodle             3             3             8
      5000000                   Sprite        Walkin             9            15             6
      5000001                    Pepci        Walkin            12            15             0
      5000002            PepciZeroLime        Walkin             7            15             0
      5000003           PepciZeroMango        Walkin            14            15            20
Item to display on the stand
Peaktime = false
      1000000            ShirimpSnack ---> 4  Urgent
```

C. 상품 판매

D. 진열 필요 상품 제시(피크타임 아닌 경우)

```
         code                    name         stand cur_stand_qnt max_stand_qnt     inventory
      1000000            ShirimpSnack         Snack             1             5            10
      1000002       PotatochipOriginal         Snack             3             5            30
      1000003          PotatochipOnion         Snack             5             5             7
      1000004           CornSnackSpicy         Snack             1             5             0
      1000005           CornSnackSweet         Snack             3             5             0
      3000000        FishcakebarSpicy  Refrigerated             5            12             0
      3000001          FishcakebarBBQ  Refrigerated             4            12             0
      3000002            ChickenChest  Refrigerated             6            12             0
      4000000      SweetSpicyCupnoodle     Cupnoodle             2             3            12
      4000001    SpicyChickenCupnoodle     Cupnoodle             3             3             8
      5000000                  Sprite        Walkin             9            15             6
      5000001                   Pepci        Walkin            12            15             0
      5000002           PepciZeroLime        Walkin             7            15             0
      5000003          PepciZeroMango        Walkin            14            15            20
Item to display on the stand
Peaktime = false
      1000000            ShirimpSnack ---> 4  Urgent
1. Sell item  2. Add item  3. Save and Exit
1
Item Code: 5000003
Quantity: 7
         code                    name         stand cur_stand_qnt max_stand_qnt     inventory
      1000000            ShirimpSnack         Snack             1             5            10
      1000002       PotatochipOriginal         Snack             3             5            30
      1000003          PotatochipOnion         Snack             5             5             7
      1000004           CornSnackSpicy         Snack             1             5             0
      1000005           CornSnackSweet         Snack             3             5             0
      3000000        FishcakebarSpicy  Refrigerated             5            12             0
      3000001          FishcakebarBBQ  Refrigerated             4            12             0
      3000002            ChickenChest  Refrigerated             6            12             0
      4000000      SweetSpicyCupnoodle     Cupnoodle             2             3            12
      4000001    SpicyChickenCupnoodle     Cupnoodle             3             3             8
      5000000                  Sprite        Walkin             9            15             6
      5000001                   Pepci        Walkin            12            15             0
      5000002           PepciZeroLime        Walkin             7            15             0
      5000003          PepciZeroMango        Walkin             7            15            20
Item to display on the stand
Peaktime = false
      1000000            ShirimpSnack ---> 4  Urgent
      5000003          PepciZeroMango ---> 8  Not Urgent
```

E. 진열 필요 상품 제시(피크 타임인 경우)

```
         code                    name         stand cur_stand_qnt max_stand_qnt     inventory
      1000000            ShirimpSnack         Snack             1             5            10
      1000002       PotatochipOriginal         Snack             3             5            30
      1000003          PotatochipOnion         Snack             5             5             7
      1000004           CornSnackSpicy         Snack             1             5             0
      1000005           CornSnackSweet         Snack             3             5             0
      3000000        FishcakebarSpicy  Refrigerated             5            12             0
      3000001          FishcakebarBBQ  Refrigerated             4            12             0
      3000002            ChickenChest  Refrigerated             6            12             0
      4000000      SweetSpicyCupnoodle     Cupnoodle             2             3            12
      4000001    SpicyChickenCupnoodle     Cupnoodle             3             3             8
      5000000                  Sprite        Walkin             9            15             6
      5000001                   Pepci        Walkin            12            15             0
      5000002           PepciZeroLime        Walkin             7            15             0
Item to display on the stand
Peaktime = true
      1000000            ShirimpSnack ---> 4
      1000002       PotatochipOriginal ---> 2
      4000000      SweetSpicyCupnoodle ---> 1
      5000000                  Sprite ---> 6
1. Sell item  2. Add item  3. Save and Exit
```

5. 결과 및 결론

A. 제안서의 기능1, 기능2-1, 기능2-2를 구현함

B. 추후 코딩하는 과정에서 구조적인 개선점이 보여 약간 보완하는 작업을 기능2-3을 구현하는 과정과 동시에 진행할 예정