รายงานโครงงาน

Inventory Management Program

จัดทำโดย

นายวิชญ์พล	วิทยโกมล	รหัส	6513120
นายศุภกร	ถาวรวงศ์	รหัส	6513121
นายอนพัทย์	กิจสมมารถ	รหัส	6513123
นายมหรรณพ	ถาบัว	รหัส	6513136
นายจิณณพัต	น้อยวัฒน์	รหัส	6513162

เสนอ

ผศ.ดร. มิ่งมานัส ศิวรักษ์

รายงานนี้เป็นส่วนหนึ่งของรายวิชาคณะวิศวกรรมศาสตร์หลักสูตรปริญญาตรี

เทคนิคการเขียนโปรแกรม (วศคพ 112)

ภาคการศึกษาภาคปลาย ปีการศึกษา 2565

มหาวิทยาลัยมหิดล

สารบัญ

1. ไอเดีย และการทำงานของโปรแกรม

1.1 ไอเดียของโปรแกรม

โปรแกรมนี้ออกแบบมาเพื่อจัดการคลังสินค้า และกำจัดตามเวลาที่เหมาะสม (หมดอายุ) โดยอ้างอิงจาก Theme : dispose โปรแกรมใช้หลักการทำงานของ Linked List ในการจัดเก็บข้อมูลของสินค้าเพื่ออ่านและแก้ไขผ่าน Console โดยใช้การประยุกต์การอ่านข้อมูลจากไฟล์เพื่อสำเนาข้อมูลมาจัดเก็บโดยใช้ Linked-List ทำให้สามารถทำการค้นหา เพิ่ม ลด แก้ไขข้อมูลได้ผ่านหน้าจอ Console จากนั้นบันทึกไฟล์กลับ หลังจากมีการแก้ไขแล้ว มีการออกแบบ Linked List โดยแบ่ง Class เพื่อกำหนดค่าที่จะจัดเก็บใน Node เช่น Class Consumable, Class Food, และ Class Drink โดย Consumable จะถ่ายทอดให้แก่ Food และ Drink โดยมีฟังก์ชันเช่น display() ในลักษณะของ polymorphism โดยกำหนด virtual ใน Food เป็นต้น จากนั้นนำมาสร้างเป็น Linked List

1.2 การทำงานของโปรแกรม

หลักการทำงานโดยสังเขป : File.txt ⟶ Linked List (Display Sort Edit) ⟶ File.txt
การทำงานของโปรแกรมแบ่งออกเป็น 3 ส่วนดังนี้

1.2.1 การเริ่มโปรแกรม

เมื่อผู้ใช้งานเปิดโปรแกรมขึ้นมาจะเข้าสู่หน้า Main menu ในหน้า Main menu จะมีตัวเลือกให้ผู้ใช้งานเลือกทั้งหมด 6 ข้อได้แก่

ตัวเลือก 1: Load Data

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะทำการโหลดข้อมูลจาก Text file เข้าสู่ Node สำหรับใช้ในการแสดงผล หรือแก้ไขข้อมูล

หลักการทำงาน : โปรแกรมจะเปิด Text file เพื่อเก็บข้อมูลของสินค้าเข้าไปใน Node แล้วเชื่อมต่อกันในรูปแบบ Linked List

ตัวเลือก 2: Show All Data

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะแสดงข้อมูลของสินค้าทั้ง 4 ประเภท ได้แก่ Food, Drink, Daily use และ Special Appliance

หลักการทำงาน : โปรแกรมจะแสดงข้อมูลของสินค้าทุกประเภทที่บันทึกอยู่ใน Node ผ่านฟังก์ชัน show_all()

ตัวเลือก 3: Add, Delete, Edit your Stock

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะเข้าสู่หน้าถัดไปเพื่อที่จะ เพิ่ม ลบ หรือ แก้ไข ข้อมูลสินค้า

ตัวเลือก 4: Sorting Your Stock

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะเข้าสู่หน้าถัดไปเพื่อที่จะเรียงลำดับตามหัวข้อต่าง ๆ สินค้า

ตัวเลือก 9: Exit

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะแสดงหน้าต่างให้ผู้ใช้งานเลือกที่จะบันทึกข้อมูลที่ได้มีการแก้ไขหรือไม่ หากผู้ใช้เลือกที่จะบันทึกโปรแกรมจะบันทึกข้อมูลใน Node ทั้งหมดทับลงใน Text file เดิม

ตัวเลือก 99: Then type: "dev" to switch developer mode

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะให้ผู้ใช้งานพิมพ์คำว่า “dev” ซึ่งเปรียบเสมือน Password สำหรับเข้าสู่ Developer mode สำหรับแสดงข้อมูลที่ละเอียดขึ้น

1.2.2 การเพิ่ม ลบ และแก้ไขข้อมูล

หลังจากที่ผู้ใช้งานเลือกตัวเลือกนี้แล้ว โปรแกรมจะแสดงหน้าตัวเลือกต่าง ๆ ได้แก่

ตัวเลือก 1: Add Stock

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะเข้าสู่หน้าต่อไปเพื่อให้เลือกหมวดหมู่ที่ผู้ใช้งานต้องการเพิ่ม โดยจะมีการแสดงข้อมูลของสินค้าทั้งหมดในหมวดหมู่ที่ผู้ใช้งานเลือกนั้นผ่านฟังก์ชัน show_all() หลังจากนั้นให้ผู้ใช้งานเลือกหมวดหมู่ที่ต้องการจะเพิ่ม หลังจากนั้นโปรแกรมจะให้ผู้ใช้งานกรอกข้อมูลสินค้าเพื่อเพิ่มสินค้าใหม่ใน Stock

<u>หลักการทำงาน</u> : เมื่อมีการเรียกฟังก์ชัน add_stock() โปรแกรมจะถามหาหมวดหมู่ที่ผู้ใช้งานต้องการจะเพิ่มโดยมีการแสดงข้อมูลสินค้าทั้งหมดในหมวดหมู่นั้นผ่านฟังก์ชัน show_all() หลังจากนั้นโปรแกรมจะเพิ่มข้อมูลใหม่ไปที่ Head of Linked List ของหมวดหมู่นั้น

ตัวเลือก 2: Delete Stock

หลักการทำงาน : เมื่อมีการเรียกฟังก์ชัน show_delete_stock()โปรแกรมจะถามหาหมวดหมู่ ที่ผู้ใช้งานต้องการจะลบโดยมีการแสดงข้อมูลสินค้าทั้งหมดในหมวดหมู่นั้นผ่านฟังก์ชัน show_all() หลังจากนั้นโปรแกรมจะลบข้อมูลของสินค้าที่ผู้ใช้งานเลือกออกจาก Linked List

ตัวเลือก 3: Edit Stock

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะเข้าสู่หน้าต่อไปเพื่อให้เลือกหมวดหมู่ที่ผู้ใช้งานต้องการจะแก้ไข หลังจากนั้นให้ผู้ใช้งานเลือกหมวดหมู่ที่ต้องการจะแก้ไขโดยจะมีการแสดงข้อมูลของสินค้าทั้งหมดในหมวดหมู่ที่ผู้ใช้งานเลือก หลังจากนั้นโปรแกรมจะให้ผู้ใช้งานกรอก NodeID ที่ต้องการจะแก้ไข หากผู้ใช้งานต้องการกลับไปที่หน้า Main menu ให้ผู้ใช้งานกรอกหมายเลย 0

หลักการทำงาน : เมื่อมีการเรียกฟังก์ชัน edit_stock() โปรแกรมจะถามหาหมวดหมู่ที่ผู้ใช้งานต้องการจะแก้ไข หลังจากนั้นโปรแกรมจะเรียกฟังก์ชัน delete_stock() เพื่อลบข้อมูลของสินค้าที่ผู้ใช้งานเลือกออกจาก Link list และเรียกฟังก์ชั่น add_stock() เพื่อเพิ่มข้อมูลใหม่เข้าไป

ตัวเลือก 4: Back to main menu : หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะสู่หน้า Main menu

1.2.3 การเรียงลำดับ (Sorting)

หลังจากที่ผู้ใช้งานเลือกตัวเลือกนี้ โปรแกรมจะเข้าสู่หน้าต่างเลือกหมวดหมู่ที่ผู้ใช้งานต้องการเรียงลำดับ ในโปรแกรมนี้ จะใช้ Insertion Sort Algorithms เป็นรูปแบบของในการเรียงข้อมูลใน linked list ข้อมูลของแต่ละคลาสจะถูกเรียงจากน้อยไปมาก โดยที่สิ่งนำมาเป็นตัวเปรียบเทียบ จะมีทั้งหมด 5 อย่าง ได้แก่

ตัวเลือก 1: Sorting by Node ID

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะเรียงลำดับข้อมูลตามลำดับ NodeID ในหมวดหมู่ที่ผู้ใช้งานเลือกหลักการทำงาน ใช้ฟังก์ชัน return_NodeID() ที่ส่งค่ากลับออกมาเป็นไอดี ของหมายเลขประจำตัวในแต่ละสินค้าเป็นตัวเปรียบเทียบในการทำงานของชุดนี้

ตัวเลือก 2: Sorting by MFD

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะเรียงลำดับข้อมูลตามลำดับวันผลิต ในหมวดหมู่ที่ผู้ใช้งานเลือกหลักการทำงาน ใช้ฟังก์ชัน return_mfg() ที่ส่งค่ากลับออกมาเป็น string ของเดือนและปีวันผลิต (mfg) และเรียกใช้ในฟังก์ชัน compareDates() ในการรับค่าเดือนและปีของสินค้าแต่ละชิ้นที่ถูกเก็บไว้ แปลงเป็น integer แล้วทำการเปรียบเทียบเดือน และวัน ตามลำดับ

ตัวเลือก 3: Sorting by EXP

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะเรียงลำดับข้อมูลตามลำดับวันหมดอายุ ในหมวดหมู่ที่ผู้ใช้งานเลือกหลักการทำงาน ใช้ฟังก์ชัน return_exp() ที่ส่งค่ากลับออกมาเป็น string ของเดือน และปีของวันหมดอายุ (exp) โดยขั้นตอนการทำเหมือนกันการเรียงโดยใช้วันผลิต (mfg)

ตัวเลือก 4: Sorting by Weight

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะเรียงลำดับข้อมูลตามลำดับวันหมดอายุ ในหมวดหมู่ที่ผู้ใช้งานเลือกหลักการทำงาน ใช้ฟังก์ชัน return_weightAll() ในการส่งค่ากลับออกมาเป็นน้ำหนักทั้งหมด ของสินค้าแต่ละชิ้นนำมาเป็นตัวเปรียบเทียบข้อมูล

ตัวเลือก 5: Go Back To Menu

หากผู้ใช้งานเลือกตัวเลือกนี้โปรแกรมจะกลับสู่หน้า Main menu

ตัวอย่างของโค้ดที่ใช้:

```
void LLFood::sorting_by_NodeID() {
    clear();

    food* current = hol;
    food* sorted = nullptr;

    if (current == nullptr || current->move_next() == nullptr)
        return;

    while (current != nullptr) {
        food* nextNode = current->move_next();
        food* prev = nullptr;
        food* temp = sorted;

        while (temp != nullptr && current->return_NodeID() > temp->return_NodeID()) {
            prev = temp;
            temp = temp->move_next();
        }

        if (prev == nullptr) {
            current->set_next(sorted);
            sorted = current;
        } else {
            prev->set_next(current);
            current->set_next(temp);
        }

        current = nextNode;
    }

    hol = sorted;
    cout << "=== Successfully Sorted === " << endl;
    a.show_all();
}
```

2) ข้อจำกัดของโปรแกรม

2.1) ไม่สามารถแก้ไขข้อมูลอย่างละเอียดของแต่ละสินค้าได้ ผู้ใช้จำเป็นต้องกรอกข้อมูลของสินค้าชนิดนั้นใหม่ทั้งหมด

2.2) รูปแบบของวันผลิต และวันหมดอายุไม่ได้มีการกำหนดที่แน่นอน หากมีการรับค่าข้อมูลในรูปแบบที่แตกต่างกัน อาจส่งผลต่อการเรียงลำดับตามวันผลิต หรือวันหมดอายุได้

2.3) ไม่ได้มีการจำกัดจำนวณตัวอักษรขณะรับค่า หากมีการรับค่าข้อมูลที่มีจำนวณตัวอักษรมากเกินไป อาจส่งผลต่อรูปแบบการแสดงผลได้

2.4) ไม่สามารถกรอกช่องว่าง ขณะรับค่าข้อมูลในรูปแบบ string ได้

3) Class

Class Node :

```
/*Node*/
class NODE {
    private:
    protected:
        long NodeID;
    public:
        NODE();
        virtual ~NODE() = 0;
        virtual void show_data() = 0;
};
```

Class General :

```
/*general*/
class general:public NODE{
    private:
    protected:
        long LotID, LotNo;
        string name;
        string mfg, exp;
        string updateTime;
    public:
        general(string = "No Name", string = "No MFG", string = "No EXP", long = 0);
        virtual ~general() = 0;
        virtual void show_data() = 0;;
};
```

Class Consumable :

```
/*consumable: general*/
class consumable:public general{
    private:
    protected:
        double weightAll;
        int quantityContainer, quantityEach;
    public:
        consumable(string = "No Name", string = "No MFG", string = "No EXP", long = 0, double = 0.0, int = 0,int = 0);
        virtual ~consumable() = 0;
        virtual void show_data() = 0;
};
```

Class Food :

```
/*food:consumable*/
class food:public consumable{
    private:
        food *next;
    protected:
        double weightEach;
    public:
        food(string = "No Name", string = "No MFG", string = "No EXP", long = 0, double = 0.0, int = 0,int = 0, double = 0.0);
        ~food();
        void show_data();
        void insert(food*&);
        food* move_next();
        int return_NodeID();
        void edit_NodeID(int);
        void set_next(food *);
        double return_weightAll();
        string return_mfg();
        string return_exp();
};
```

Class Drinks :

```
/*drinks: consumable*/
class drinks:public consumable{
    private:
        drinks *next;
    protected:
        double volumeEach;
    public:
        drinks(string = "No Name", string = "No MFG", string = "No EXP", long = 0, double = 0.0, int = 0,int = 0, double = 0.0);
        ~drinks();
        void show_data();
        void insert( drinks*&);
        drinks* move_next();
        //Son code
        int return_NodeID();
        void edit_NodeID(int);
        void set_next(drinks *);
        double return_weightAll();
        double return_volumeAll();
        string return_mfg();
        string return_exp();
        //long re_NodeID();
};
```

Class Appliance :

```
/*appliance: general*/
class appliance:public general{
    private:
    protected:
        string Usefor;
        double weightAll;
        int quantityContainer, quantityEach;
    public:
        appliance(string = "No Name", string = "No MFG", string = "No EXP", long = 0, string = "For non", double = 0.0, int = 0, int = 0);
        virtual ~appliance() = 0;
        virtual void show_data() = 0;
        appliance* move_next();
        //Son code
        int return_NodeID();
        void edit_NodeID(int);
        //long re_NodeID();
};
```

Class DailyUse :

```
/*dailyUse: appliance*/
class dailyUse:public appliance{
    private:
        dailyUse *next;
    protected:
        string details;
    public:
        dailyUse(string = "No Name", string = "No MFG", string = "No EXP", long = 0, string = "For non", double = 0.0, int = 0, int = 0, string = "No detail");
        ~dailyUse();
        void show_data();
        void insert(dailyUse *&);
        dailyUse* move_next();
        //Son code
        int return_NodeID();
        void edit_NodeID(int);
        void set_next(dailyUse *);
        double return_weightAll();
        string return_mfg();
        string return_exp();
        //long re_NodeID();
};
```

Class SpecificUse :

```
/*specificPurpose: appliance*/
class specificPurpose:public appliance{
    private:
        specificPurpose *next;
    protected:
        string caution;
    public:
        specificPurpose(string = "No Name", string = "No MFG", string = "No EXP", long = 0, string = "For non", double = 0.0, int = 0, int = 0, string = "No caution");
        ~specificPurpose();
        void show_data();
        void insert( specificPurpose*&);
        specificPurpose* move_next();
        //Son code
        int return_NodeID();
        void edit_NodeID(int);
        void set_next(specificPurpose *);
        double return_weightAll();
        string return_mfg();
        string return_exp();
        //long re_NodeID();
};
```

4) Requirement :

4.1) Class Constructor & Polymorphism :

Class NODE :

```
/*Node*/
NODE::NODE(){
      static long x = 130000;
      NodeID = ++x;
      if(dev){
      cout << "adding NodeID @ NODE : " << x << endl;
      }
}
 NODE:: ~NODE(){
      if(dev) {
      cout << "NodeID " << NodeID << " is being deleted" << endl;
      }
}
void NODE:: show_data(){
     cout << "NodeID @ NODE : " << NodeID << endl;
}
```

Class General (Inherited from NODE class) :

```
general::general(string inName, string inmfg, string inexp, long inLotNo):NODE(){
      static long x = 0, y = 500000;
      LotID = ++x;
      if(inLotNo) LotNo = ++y; // Default inLotNo should be 0
      else LotNo = inLotNo;
      name = inName;
      mfg = inmfg;
      exp = inexp;

      // time input
      time_t curr_time;
      curr_time = time(NULL);
      string realtime = ctime(&curr_time);
      string textDay, day, textMonth, year, hh, mm, ss, time;
      if(dev) cout << "Today is : " << realtime; //In this format >> Tue Apr 25 12:31:25 2023
      textDay = textDay + realtime[0] + realtime[1] + realtime[2]; // Tue
      day = day + realtime[8] + realtime[9]; // 25
      textMonth = textMonth + realtime[4] + realtime[5] + realtime[6]; // Apr
      year = year + realtime[20] + realtime[21] + realtime[22] + realtime[23]; //2023
      hh = hh + realtime[11] + realtime[12]; //12
      mm = mm + realtime[14] + realtime[15]; //31
      ss = ss + realtime[17] + realtime[18]; //25
      time = time + hh + ":" + mm + ":" + ss; // hh:mm:ss
      updateTime = updateTime + " " + textDay + " " +  day + " " +  textMonth + " " +  year + " @ " +  time; // Tue 25 Apr 2023 12:31:25

      if(dev){
      cout << "textDay :" << textDay << endl;
      cout << "day :" << day << endl;
      cout << "textMonth :" << textMonth << endl;
      cout << "year :" << year << endl;
      cout << "hh :" << hh << endl;
      cout << "mm :" << mm << endl;
      cout << "ss :" << ss << endl;
      cout << "time :" << time << endl;
      cout << "adding name @ general : " << name << endl;
      cout << "adding mfg @ general : " << mfg << endl;
      cout << "adding exp @ general : " << exp << endl;
      cout << "adding updateTime @ general : " << updateTime << endl;
      }
```

Class Consumable (Inherited from General Class) :

```
/*consumable: general*/
consumable::consumable(string inName, string inmfg, string inexp, long inLotNo, double inweightAll, int inquantityContainer, int inquantityEach):general(inName, inmfg, inexp, inLotNo){
      weightAll = inweightAll;
      quantityContainer = inquantityContainer;
      quantityEach = inquantityEach;
      if(dev){
      cout << "adding weightAll @ consumable : " << weightAll << endl;
      cout << "adding quantityContainer @ consumable : " << quantityContainer << endl;
      cout << "adding quantityEach @ consumable : " << quantityEach << endl;
      }
}
consumable:: ~consumable(){
      if(dev){
      cout << "weightAll " << weightAll << " is being deleted" << endl;
      cout << "quantityContainer " << quantityContainer << " is being deleted" << endl;
      cout << "quantityEach " << quantityEach << " is being deleted" << endl;
      }
}
void  consumable:: show_data(){
      cout << "weightAll @ consumable : " << weightAll << endl;
      cout << "quantityContainer @ consumable : " << quantityContainer << endl;
      cout << "quantityEach @ consumable : " << quantityEach << endl;
}
```

Class Food (Inherited from Consumable Class) :

```
/*food:consumable*/
food::food(string inName, string inmfg, string inexp, long inLotNo, double inweightAll, int inquantityContainer, int inquantityEach, double inweightEach):consumable(inName, inmfg, inexp, inLotNo, inweightAll, inquantityContainer, inquantityEach){
      weightEach = inweightEach;
      if(dev){
      cout << "adding weightEach @ food : " << weightEach << endl;
      }
}
food:: ~food(){
      if(dev){
      cout << "weightEach " << weightEach << " is being deleted" << endl;
      }
}
void food:: show_data(){
      NODE::show_data();
      general::show_data();
      consumable::show_data();
      cout << "weightEach @ food : " << weightEach << endl;
}
food *food::move_next(){
      return next;
}

void food::insert(food*& x){
     x->next=this;
}
```

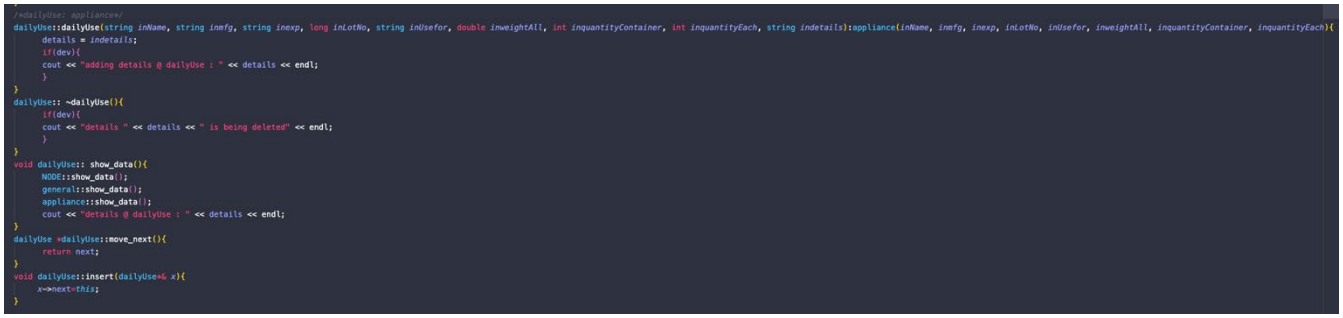
Class Drinks (Inherited from Consumable Class) :

```
/*drinks: consumable*/
drinks::drinks(string inName, string inmfg, string inexp, long inLotNo, double inweightAll, int inquantityContainer, int inquantityEach, double involumeEach):consumable(inName, inmfg, inexp, inLotNo, inweightAll, inquantityContainer, inquantityEach){
      volumeEach = involumeEach;
      if(dev){
      cout << "adding volumeEach @ drinks : " << volumeEach << endl;
      }
}
drinks:: ~drinks(){
      if(dev){
      cout << "volumeEach " << volumeEach << " is being deleted" << endl;
      }
}
void drinks:: show_data(){
      NODE::show_data();
      general::show_data();
      consumable::show_data();
      cout << "volumeEach @ drinks : " << volumeEach << endl;
}
drinks *drinks::move_next(){
      return next;
}
void drinks::insert(drinks*& x){
     x->next=this;
}
```

Class Appliance (Inherited from General Class) :

```
/*appliance: general*/
appliance::appliance(string inName, string inmfg, string inexp, long inLotNo, string inUsefor, double inweightAll, int inquantityContainer, int inquantityEach):general(inName, inmfg, inexp, inLotNo){
      Usefor = inUsefor;
      weightAll = inweightAll;
      quantityContainer = inquantityContainer;
      quantityEach = inquantityEach;
      if(dev){
      cout << "adding Usefor @ appliance : " << Usefor << endl;
      cout << "adding weightAll @ appliance : " << weightAll << endl;
      cout << "adding quantityContainer @ appliance : " << quantityContainer << endl;
      cout << "adding quantityEach @ appliance : " << quantityEach << endl;
      }
}
appliance:: ~appliance(){
      if(dev){
      cout << "Usefor " << Usefor << " is being deleted" << endl;
      cout << "weightAll " << weightAll << " is being deleted" << endl;
      cout << "quantityContainer " << quantityContainer << " is being deleted" << endl;
      cout << "quantityEach " << quantityEach << " is being deleted" << endl;
      }
}
void  appliance:: show_data(){
      cout << "Usefor @ appliance : " << Usefor << endl;
      cout << "weightAll @ appliance : " << weightAll << endl;
      cout << "quantityContainer @ appliance : " << quantityContainer << endl;
      cout << "quantityEach @ appliance : " << quantityEach << endl;
}
```

Class DailyUse (Inherited from Appliance Class) :



```
/*dailyUse: appliance*/
dailyUse::dailyUse(string inName, string inmfg, string inexp, long inLotNo, string inUsefor, double inweightAll, int inquantityContainer, int inquantityEach, string indetails):appliance(inName, inmfg, inexp, inLotNo, inUsefor, inweightAll, inquantityContainer, inquantityEach){
      details = indetails;
      if(dev){
      cout << "adding details @ dailyUse : " << details << endl;
      }
}
dailyUse:: ~dailyUse(){
      if(dev){
      cout << "details " << details << " is being deleted" << endl;
      }
}
void dailyUse:: show_data(){
      NODE::show_data();
      general::show_data();
      appliance::show_data();
      cout << "details @ dailyUse : " << details << endl;
}
dailyUse *dailyUse::move_next(){
      return next;
}
void dailyUse::insert(dailyUse*& x){
      x->next=this;
}
```

Class SpecificUse (Inherited from Appliance Class) :

```
/*specificPurpose: appliance*/
specificPurpose::specificPurpose(string inName, string inmfg, string inexp, long inLotNo, string inUsefor, double inweightAll, int inquantityContainer, int inquantityEach, string incaution):appliance(inName, inmfg, inexp, inLotNo, inUsefor, inweightAll, inquantityContainer, inquantityEach){
      caution = incaution;
      if(dev){
      cout << "adding caution @ specificPurpose : " << caution << endl;
      }
}
specificPurpose:: ~specificPurpose(){
      if(dev){
      cout << "caution " << caution << " is being deleted" << endl;
      }
}
void  specificPurpose:: show_data(){
      NODE::show_data();
      general::show_data();
      appliance::show_data();
      cout << "caution @ specificPurpose : " << caution << endl;
}
specificPurpose *specificPurpose::move_next(){
      return next;
}
void specificPurpose::insert(specificPurpose*& x){
      x->next=this;
}
```

4.2) Data Structure (Linked List) :

Class Linked List of Food :

```
/*LLFood*/
class LLFood{
      food *hol;
      int size;
public:
       void add_node(food *&);
       void show_all();
       ~LLFood();
       LLFood();
       bool check_NodeID(int);
       void delete_stock(int);
       int LLreturn_NodeID(int);
       void LLedit_NodeID(int);
       void sorting_by_NodeID();
       void sorting_by_mfd();
       void sorting_by_exp();
       void sorting_by_weight();
};
```

Class Linked List of Drinks :

```
/*LLDrinks*/
class LLDrinks{
      drinks *hol;
      int size;
public:
       void add_node(drinks*&);
       void show_all();
       ~LLDrinks();
       LLDrinks();
       //Son code
       bool check_NodeID(int);
       void delete_stock(int);
       int LLreturn_NodeID(int);
       void LLedit_NodeID(int);
       void sorting_by_NodeID();
       void sorting_by_mfd();
       void sorting_by_exp();
       void sorting_by_weight();
       void sorting_by_volume();
};
```

Class Linked List of Daily Food :

```
/*LLDai*/
class LLDai{
      dailyUse *hol;
      int size;
public:
       void add_node(dailyUse *&);
       void show_all();
       ~LLDai();
       LLDai();
       //Son code
       bool check_NodeID(int);
       void delete_stock(int);
       int LLreturn_NodeID(int);
       void LLedit_NodeID(int);
       void sorting_by_NodeID();
       void sorting_by_mfd();
       void sorting_by_exp();
       void sorting_by_weight();
};
```

Class Linked List of Specific Use :

```
/*LLSpec*/
class LLSpec{
      specificPurpose *hol;
      int size;
public:
       void add_node(specificPurpose *&);
       void show_all();
       ~LLSpec();
       LLSpec();
       //Son code
       bool check_NodeID(int);
       void delete_stock(int);
       int LLreturn_NodeID(int);
       void LLedit_NodeID(int);
       void sorting_by_NodeID();
       void sorting_by_mfd();
       void sorting_by_exp();
       void sorting_by_weight();
};
```

4.3) Exception Handling : ฟังก์ชัน scin() และ ask() ในโปรแกรมช่วยในการรับค่าจาก User ทั้งหมด

4.3.1) ฟังก์ชัน scin() : จะแบ่งเป็น 2 ประเภทคือ

4.3.1.1) scin((type) &choice) ที่ใช้ใน try{} ซึ่งจะเช็คการกรอกที่ผิดพลาดและ throw อัตโนมัติ

4.3.1.2) scin((type) &choice, int &choice_check) ที่ออกแบบมาให้ใช้กับ ask()

4.3.2) ฟังก์ชัน ask() : ฟังก์ชันที่จะรับค่า พร้อมแสดงคำถามให้ผู้ใช้เห็น และตรวจสอบการใส่ค่าอัตโนมัติ โดยไม่ต้องใช้ try{} ก่อนการเรียกใช้ฟังก์ชันนี้

4.3.3) ฟังก์ชันถูกใช้ในโปรแกรมในเหตุการณ์ดังนี้

4.3.3.1) ใช้ทดแทนการรับค่าผ่าน cin เพื่อเป็นตัวเลือกใน switch() จะใช้ scin() ภายใน try{} เพื่อให้สามารถวนลูปกลับมาเลือกตัวเลือกทั้งหมดได้

4.3.3.2) ใช้ทดแทนการรับค่าผ่าน cin เพื่อรับค่าเข้าสู่ตัวแปรข้อมูลโดยเฉพาะ เช่น ถาม NodeID, รับค่า Name เป็นต้น จะใช้ ask() เพื่อ บังคับให้กรอก cin จนกว่าจะผ่านแล้วจึงทำคำสั่งต่อไป

ตัวอย่าง scin() :

```
//Secured cin
void scin(int &choice){
    cin >> choice;
    if(cin.fail()) throw 0;
}
void scin(string &choice){
    cin >> choice;
    if(cin.fail()) throw 0;
}
void scin(long &choice){
    cin >> choice;
    if(cin.fail()) throw 0;
}
void scin(double &choice){
    cin >> choice;
    if(cin.fail()) throw 0;
}
void scin(char &choice){
    cin >> choice;
    if(cin.fail()) throw 0;
}
```

ตัวอย่าง scin() ที่ออกแบบมาใช้กับ ask() :

```
//Secured cin with Check
void scin(int &choice, int &choice_check){
    try{
    cin >> choice;
    if(cin.fail()) {
        throw 0;
    }
    }catch( exception &e ){
        cin.clear();
        cin.ignore(100,'\n');
        if(dev) cout << "[Alert!] " << e.what() << endl;
        else cout << "[!] Input format error please try again." << endl;
        choice_check = 1;
           }
    catch(...){
        cin.clear();
        cin.ignore(100,'\n');
        if(dev) cout << "[Alert!] " << "General Exception" << endl;
        else cout << "[!] Input format error please try again." << endl;
        choice_check = 1;
    }
}
```

ตัวอย่าง ask() :

```
//Ask User
void ask(int &choice, string text){
    int ask_flag;
    do{
        ask_flag = 0;
        cout << text << " : ";
        scin(choice,ask_flag);
    }while(ask_flag == 1);
}
void ask(string &choice, string text){
    int ask_flag;
    do{
        ask_flag = 0;
        cout << text << " : ";
        scin(choice,ask_flag);
    }while(ask_flag == 1);
}
void ask(long &choice, string text){
    int ask_flag;
    do{
        ask_flag = 0;
        cout << text << " : ";
        scin(choice,ask_flag);
    }while(ask_flag == 1);
}
void ask(double &choice, string text){
    int ask_flag;
    do{
        ask_flag = 0;
        cout << text << " : ";
        scin(choice,ask_flag);
    }while(ask_flag == 1);
}
void ask(char &choice, string text){
    int ask_flag;
    do{
        ask_flag = 0;
        cout << text << " : ";
        scin(choice,ask_flag);
    }while(ask_flag == 1);
}
```