```
/* ----------------------------------------------------------------
// CFarmGame.h
// ---------------------------------------------------------------- */

#include"cworldmap.h"
#include"cworldplayer.h"
#ifndef CFARMGAME_H
#define CFARMGAME_H
class CFarmGame
{
    public:
        CFarmGame();
        CFarmGame(const CFarmGame &);
        CFarmGame & operator = (const CFarmGame &rhs);
        ~CFarmGame();
        const void Instruction(const int);
        void Start();
    private:
        CWorldMap worldmap_;
        CWorldPlayer worldplayer_;
};
#endif
/* ----------------------------------------------------------------
// CFarmGame.cpp
// ---------------------------------------------------------------- */

#include"cfarmgame.h"
#include<iostream>
#include<cstdlib>
#include<ctime>
#include<string>
using namespace std;

CFarmGame::CFarmGame()
{

}
```

```
CFarmGame::~CFarmGame(){}

const void CFarmGame::Instruction(const int i){
     cout << worldplayer_.playerlist_[i].GetName() << ", 請選擇? (1:擲骰子/ 2:離開遊戲)...> ";
}

void CFarmGame::Start()
{
     int person = 0;
     const int ini_money = 5000;

     cout << "請輸入玩家總數?(Maximum:4)...> ";
     cin >> person;
     worldplayer_.SetPerson(person);
     worldmap_.IniRoad(person);
     for(int i = 0 ; i < person ; i++)
     {
          string name;
          cout << "請輸入玩家"<< i + 1 << " 的名字: ";
          cin >> name;
          worldplayer_.playerlist_[i].SetName(name);
          worldplayer_.playerlist_[i].SetLocation(0);
          worldplayer_.playerlist_[i].SetMoney(ini_money);
          worldplayer_.playerlist_[i].SetIdentifier(i);
     }

     worldmap_.ReadMapData();
     int currentplayer = 0;
     bool gamestart = true;
     while(gamestart)
     {
          srand((unsigned)time(NULL));
          system("cls");
          worldmap_.ShowAll();
          worldplayer_.ShowAll(currentplayer);
          Instruction(currentplayer);
          int steps = rand() % 6 + 1;
          char choice = '0' ;
```

```
        while(choice != '2')
        {
            cin.get(choice) >> choice;
            switch(choice){
                case '1':
                    choice = '2';
                    break;
                case '2':
                    choice = '2';
                    gamestart = false;
                    break;
                default:
                    cout << "輸入錯誤, 請重新輸入!" << endl;
                    break;
            }
        }

        if(gamestart){
            worldmap_.SetRoad(worldplayer_.playerlist_[currentplayer] ,steps);
            system("cls");
            worldmap_.ShowAll();
            worldplayer_.ShowAll(currentplayer);
            worldmap_.EveryTurn();

    worldmap_.map_[worldplayer_.playerlist_[currentplayer].GetLocation()]->PlayerVisit(worldpl
ayer_.playerlist_[currentplayer]);
            currentplayer = ( currentplayer + 1 ) % person;
            system("pause");
        }

        if(worldplayer_.playerlist_[currentplayer].GetMoney() <= 0){
            cout << worldplayer_.playerlist_[currentplayer].GetName()
                << "破產了, 所擁有的東西將全數歸還...." << endl;
            system("pause");
            worldmap_.RIP(currentplayer);
            worldplayer_.PlayerDie(currentplayer);
        }
        int dieman = 0;
```

```
        for(int i = 0 ; i < person ; i++){
            if(worldplayer_.playerlist_[i].IsDie()){
                dieman++;
            }
        }

        if(dieman == person){
            gamestart = false;
            cout << "全數玩家都破產了, 遊戲結束...." << endl;
            system("pause");
        }
    }
}

/* ----------------------------------------------------------------
// CPlayer.h
// ---------------------------------------------------------------- */

#include <string>
using namespace std;
#ifndef CPLAYER_H
#define CPLAYER_H
class CPlayer
{
    public:
        CPlayer() : identifier _(0), location _(0), money _(0), owned_(0), name_(""),
isdie(false){};
        CPlayer(const CPlayer &rhs);
        CPlayer & operator = (const CPlayer &rhs);
        ~CPlayer();
        const string GetName();
        const int GetLocation();
        const int GetMoney();
        void AddMoney(const int);
        const int GetIdentifier();
        const int GetOwned();
        void SetName(const string );
        void SetLocation(const int );
```

```
        void SetMoney(const int );
        void SetIdentifier(const int );
        void SetOwned(const int );
        void Die();
        bool IsDie();
    private:
        string name_;
        int identifier_, location_, money_, owned_;
        bool isdie_;
};
#endif
/* ----------------------------------------------------------------
// CPlayer.cpp
// ---------------------------------------------------------------- */

#include<string>
#include"cplayer.h"
using namespace std;

CPlayer::~CPlayer(){}

void CPlayer::Die(){isdie_ = true;}
void CPlayer::SetName(const string name){name_ = name;}

const std::string CPlayer::GetName(){return name_;}
void CPlayer::SetLocation(const int location){location_ = location;}
const int CPlayer::GetLocation(){return location_;}
void CPlayer::SetMoney(int money){money_ = money;}
const int CPlayer::GetMoney(){return money_;}
void CPlayer::AddMoney(const int cash){money_ += cash;}
void CPlayer::SetIdentifier(const int identifier){identifier_ = identifier;}
const int CPlayer::GetIdentifier(){return identifier_;}
bool CPlayer::IsDie(){return isdie_;}
const int CPlayer::GetOwned(){return owned_;}
void CPlayer::SetOwned(const int owned){owned_ = owned;}
```

```
/* ----------------------------------------------------------------
// CWorldPlayer.h
// ---------------------------------------------------------------- */

#include"cplayer.h"
#ifndef CWORLDPLAYER_H
#define CWORLDPLAYER_H
class CWorldPlayer
{
    public:
        CWorldPlayer();
        CWorldPlayer(const CWorldPlayer &);
        CWorldPlayer & operator = (const CWorldPlayer &rhs);
        ~CWorldPlayer();
        void SetPerson(int);
        const void ShowAll(int);
        void PlayerDie(int);
        friend class CFarmGame;
    protected:
        CPlayer *playerlist_;
        int person_;
};
#endif

/* ----------------------------------------------------------------
// CWorldPlayer.cpp
// ---------------------------------------------------------------- */

#include"CWorldPlayer.h"
#include<iostream>
#include<iomanip>
using namespace std;

CWorldPlayer::CWorldPlayer(){}

CWorldPlayer::~CWorldPlayer(){delete []playerlist_;}
```

```
void CWorldPlayer::SetPerson(int people)
{
     person_ = people;
     playerlist_ = new CPlayer[person_];
}

void CWorldPlayer::PlayerDie(int player)
{
     playerlist_[player].Die();
}

const void CWorldPlayer::ShowAll(int current)
{
     for(int i = 0 ; i < person_ ; i++)
     {

          if(!playerlist_[i].IsDie())
          {
               if(i == current)
               {
                    cout << "=>[";
               }
               else
               {
                    cout << "  [";
               }
               cout << i << "]"<< setw(20) << playerlist_[i].GetName() << setw(4) << "$" << setw(6)
<< playerlist_[i].GetMoney() << " 擁有" << playerlist_[i].GetOwned() << " 單位" << endl;
          }
     }
     cout << endl;
}
/*
=>[0]           阿土伯  $5000 擁有0 單位
  [1]           孫小美  $5000 擁有0 單位
  [2]           金貝貝  $5000 擁有0 單位
```

```
*//* ---------------------------------------------------------------------
// CWorldMap.h
// --------------------------------------------------------------------- */

#include"cmapunit.h"
#include"cvegetableunit.h"
#include"corchardunit.h"
#include"clivestockunit.h"
#include"cpoultryunit.h"
#ifndef CWORLDMAP_H
#define CWORLDMAP_H
class CWorldMap
{
    public:
        CWorldMap();
        CWorldMap(const CWorldMap &);
        CWorldMap & operator = (const CWorldMap &rhs);
        ~CWorldMap();
        int GetMapSize();
        void ReadMapData();
        void IniRoad(int);
        void SetRoad(CPlayer &, int);
        const void ShowAll();
        void EveryTurn();
        void RIP(int);
        friend class CFarmGame;
    protected:
        CMapUnit **map_;
        bool **playerroad_;
        int mapsize_, person_;
};
#endif
```

```
/* ----------------------------------------------------------------------
// CWorldMap.cpp
// ---------------------------------------------------------------------- */

#include"cworldmap.h"
#include <string>
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

CWorldMap::CWorldMap(){
     ReadMapData();
}

CWorldMap::~CWorldMap()
{
     for(int i = 0 ; i < mapsize_ ; i++)
     {
          delete map_[i];
     }
     delete map_;
}

int CWorldMap::GetMapSize()
{
     return mapsize_;
}
```

```
/*
 * ReadMapData : read board from a saved file
 * 檔案內含: 地圖資料
 */
void CWorldMap::ReadMapData()
{
     ifstream fptr ("farmmap.farmmap", ifstream::in);
     if(fptr.is_open())
     {
          fptr >> mapsize_;
          map_ = new CMapUnit*[mapsize_];
          fptr.ignore();

          for(int i = 0 ; i < mapsize_ ; i++)
          {
               char kind;
               string name, unit;
               int buy, seed, harv, period;
               fptr >> kind;
               switch(kind)
               {
                    case 'V': /* Vegetable */
                         map_[i] = new CVegetableUnit();
                         break;
                    case 'O': /* Orchard */
                         map_[i] = new COrchardUnit();
                         break;
                    case 'L': /* Livestock */
                         map_[i] = new CLivestockUnit();
                         break;
                    case 'P': /* Poultry */
                         map_[i] = new CPoultryUnit();
                         break;
                    default:
                         break;
               }
               map_[i]->SetType(kind);
               fptr >> name;
```

```
                map_[i]->SetMapName(name);
                fptr >> buy;
                map_[i]->SetBuyPrice(buy);
                fptr >> seed;
                map_[i]->SetSeedingPrice(seed);
                fptr >> harv;
                map_[i]->SetHarvestPrice(harv);
                if(kind == 'L')
                {
                     fptr >> period;
                     ((CLivestockUnit*)map_[i])->SetLivePeriod(period);
                     fptr >> unit;
                     ((CLivestockUnit*)map_[i])->SetUnit(unit);
                }
                fptr.ignore();
           }

           fptr.close();
     }
     else {
           cout << "File not exist." << endl;
     }
}

/*
 * IniRoad : initialize the player' s location
 */
void CWorldMap::IniRoad(int person){
     person_ = person;
     playerroad_ = new bool*[mapsize_];
     for(int i = 0 ; i < mapsize_ ; i++){
           playerroad_[i] = new bool[person_];
           for(int j = 0 ; j < person_ ; j++){
                if(i == 0){
                     playerroad_[i][j] = true; /* player at here */
                }
                else {    playerroad_[i][j] = false;   }    }    }    }
```

```
/*
 * SetRoad : player move.
 */
void CWorldMap::SetRoad(CPlayer &person, int steps)
{
     playerroad_[person.GetLocation()][person.GetIdentifier()] = false;
     int next = person.GetLocation() + steps;
     if(next >= mapsize_) /* if player walk back start point */
     {
          next -= mapsize_;
          person.AddMoney(-500);
     }
     person.SetLocation(next);
     playerroad_[person.GetLocation()][person.GetIdentifier()] = true;
}

/*
 * ShowAll : print all the map status
 */
const void CWorldMap::ShowAll()
{
     for(int i = 0 ; i < mapsize_ ; i++)
     {
          cout << "=";
          for(int j = 0 ; j < person_ ; j++)
          {
               if(playerroad_[i][j])
               {
                    cout << j;
               }
               else
               {
                    cout << ' ';
               }
          }
          cout << "=";
          cout << "[" << setw (2) << i << "]";
          cout << setw(10) << map_[i]->GetMapName();
```

```
        if(map_[i]->GetOwner() == -1)
        {
            cout << " {"  << ' ' << "} ";
            cout << "(-" << setw(4) << map_[i]->GetBuyPrice() << "/-" << setw(4) <<
map_[i]->GetSeedingPrice() << "/+" << setw(4) << map_[i]->GetHarvestPrice() << ")" << endl;
        }
        else
        {
            cout << " {"  << map_[i]->GetOwner() << "} ";
            cout << "(-" << setw(4) << map_[i]->GetSeedingPrice() << "/+" << setw(4) <<
map_[i]->GetHarvestPrice() << ")" ;
            switch(map_[i]->GetType())
            {
                case 'V':
                    if(((CVegetableUnit*)map_[i])->GetVGState()){cout << " 待收成" << endl;}
                    else {cout << " 未播種" << endl;  }
                    break;
                case 'O':
                    switch(((COrchardUnit*)map_[i])->GetState())                    {
                        case 0:
                            cout << " 未播種";
                            break;
                        case 1:
                            cout << " 小幼苗";
                            break;
                        case 2:
                            cout << " 小果樹";
                            break;
                        case 3:
                            cout << " 大果樹";
                            break;
                        case 4:
                            cout << " 待收成";
                            break;
                        default:
                            break;
                    }
```

```
                        for(int j = 0 ; j < ((COrchardUnit*)map_[i])->GetDamege() ; j++)    {
                             cout << "蟲";
                        }
                        cout << endl;
                        break;
                   case 'L':
                        cout << " 目前有" << setw(4) << ((CLivestockUnit*)map_[i])->GetNumber() <<
" " << ((CLivestockUnit*)map_[i])->GetUnit() << "(" << ((CLivestockUnit*)map_[i])->GetLivePeriod()
- ((CLivestockUnit*)map_[i])->GetTurn() << ")" << endl;
                        break;
                   case 'P':
                        cout << " 目前有" << setw(4) << ((CPoultryUnit*)map_[i])->GetEggNumber()
<< " 顆蛋" << endl;
                        break;
                   default:
                        break;
              }
         }
    }
    cout << endl;}
/*
= = [ 0]      菠菜園{ }  (- 100/-  10/+  60)
= = [ 1]        雞舍{ }  (- 500/-  10/+   5)
= = [ 2]    空心菜園{ }  (- 100/-  10/+  50)
= = [ 3]        馬廄{ }  (-1000/- 500/+ 100)
= = [ 4]      蘋果園{ }  (- 400/- 100/+2000)
= = [ 5]    紅蘿蔔園{ }  (- 100/-  10/+  30)
=0= [ 6]        鴨舍{ }  (- 400/-  10/+   8)
= = [ 7]        牛棚{ }  (- 800/- 400/+  80)
= = [ 8]      西瓜園{ }  (- 300/-  50/+1000)
= = [ 9]    哈蜜瓜園{ }  (- 600/- 150/+3000)
= = [10]        鵝舍{ }  (- 500/-  10/+  10)
= = [11]        驢廄{ }  (- 700/- 350/+  70)
= = [12]      香菜園{ }  (- 100/-  10/+  70)
= = [13]        鴿舍{ }  (- 400/-  10/+   3)
= = [14]      水梨園{ }  (- 700/- 200/+4000)
= = [15]        羊圈{ }  (- 600/- 300/+  60)
*/
```

```
/*
 * EveryTurn : every turn' s change of whole map
 */
void CWorldMap::EveryTurn(){
     for(int i = 0 ; i < mapsize_ ; i++){
          map_[i]->Turn();
     }
}
/*
 * RIP : when player no money, its all resource will be reset
 */
void CWorldMap::RIP(int player){
     for(int i = 0 ; i < mapsize_ ; i++){
          if(map_[i]->GetOwner() == player){
               map_[i]->Reset();
          }
     }
}/* ----------------------------------------------------------------
// CVegetableUnit.h
// ---------------------------------------------------------------- */
#include"cmapunit.h"
#ifndef CVEGETABLEUNIT_H
#define CVEGETABLEUNIT_H
class CVegetableUnit : public CMapUnit
{
     public:
          CVegetableUnit() : vg_state_(false){};
          CVegetableUnit(const CVegetableUnit &rhs);
          CVegetableUnit & operator = (const CVegetableUnit &rhs);
          virtual ~CVegetableUnit();
          virtual void PlayerVisit(CPlayer &);
          virtual void Turn();
          virtual void Reset();
          const bool GetVGState();
     private:
          bool vg_state_; /* false : can't harvest, true : can harvest */
};
#endif
```

```
/* ----------------------------------------------------------------------
// CVegetableUnit.cpp
// ---------------------------------------------------------------------- */

#include"cvegetableunit.h"
#include<iostream>
using namespace std;

CVegetableUnit::~CVegetableUnit(){}

const bool CVegetableUnit::GetVGState(){
    return vg_state_;
}
/*
 * PlayerVisit : when player visit this unit
 */
void CVegetableUnit::PlayerVisit(CPlayer &player)
{
    if(player.GetIdentifier() == owner_)
    {
        if(vg_state_) /* can harvest */
        {
            cout << "harvest, add money " << harvest_price_ << endl;
            player.AddMoney(harvest_price_);
            vg_state_ = false;
        }
        else /* can't harvest */
        {
            cout << "Do you want to SEEDING this Unit?(y/n) ";
            char choice;
            cin >> choice;
            if(choice == 'y')           {
                player.AddMoney(-1 * seeding_price_);
                vg_state_ = true;
                cout << "Spend " << seeding_price_ << " for SEEDING..... ";
            }
        }
    }
```

```
    else if(owner_ == -1) /* no owned */
    {
        cout << "Do you want to buy this Unit?(y/n) ";
        char choice;
        cin >> choice;
        if(choice == 'y')
        {
            player.AddMoney(-1 * buy_price_);
            owner_ = player.GetIdentifier();
            player.SetOwned(player.GetOwned() + 1);
            cout << "Spend " << buy_price_ << " for BUY this Unit..... ";
        }
    }
    else /* others owned */
    {   cout << "nothing to happen...." << endl;   }
}
/*
 * Turn : every turn change on this unit
 */
void CVegetableUnit::Turn(){}
/*
 * Reset : when owner is break, this unit will reset
 */
void CVegetableUnit::Reset(){
    owner_ = -1;
    vg_state_ = false;
}/* ----------------------------------------------------------------
// CPoultryUnit.h
// ---------------------------------------------------------------- */

#include"cmapunit.h"
#ifndef CPOULTRYUNIT_H
#define CPOULTRYUNIT_H
class CPoultryUnit : public CMapUnit
{
    public:
        CPoultryUnit() : egg_number_(0){};
        CPoultryUnit(const CPoultryUnit &rhs);
```

```
        CPoultryUnit & operator = (const CPoultryUnit &rhs);
        virtual ~CPoultryUnit();
        virtual void PlayerVisit(CPlayer &);
        virtual void Turn();
        virtual void Reset();
        const int GetEggNumber();
    private:
        int egg_number_;
};
#endif

/* ----------------------------------------------------------------
// CPoultryUnit.cpp
// ---------------------------------------------------------------- */

#include"cpoultryunit.h"
#include<iostream>
using namespace std;

CPoultryUnit::~CPoultryUnit(){}

const int CPoultryUnit::GetEggNumber()
{    return egg_number_;     }
/*
 * PlayerVisit : when player visit this unit
 */
void CPoultryUnit::PlayerVisit(CPlayer &player){
    if(player.GetIdentifier() == owner_){
        if(egg_number_ == 0) /* can't harvest */   {
            cout << "SEEDING~~" << endl;
            player.AddMoney(-1 * seeding_price_);
        }
        else /* can harvest */  {
            cout << "harvest, add money " << harvest_price_ * egg_number_ << endl;
            player.AddMoney(harvest_price_ * egg_number_);
            egg_number_ = 0;
        }
    }
```

```
    else if(owner_ == -1) /* no owned */{
        cout << "Do you want to buy this Unit?(y/n) ";
        char choice;
        cin >> choice;
        if(choice == 'y'){
            player.AddMoney(-1 * buy_price_);
            owner_ = player.GetIdentifier();
            player.SetOwned(player.GetOwned() + 1);
            cout << "Spend " << buy_price_ << " for BUY this Unit..... ";
        }
    }
    else /* others owned */
    {
        int pass = rand() % 2;
        if(pass == 1) /* steal success */ {
            cout << "be stool.... decrease " << egg_number_ / 2;
            egg_number_ /= 2;
        }
        else /* steal failure*/ {
            cout << "nothing to happen...." << endl;
        }
    }
}

/*
 * Turn : every turn change on this unit
 */
void CPoultryUnit::Turn(){
    if(owner_ != -1)
    {    egg_number_++;     }
}
/*
 * Reset : when owner is break, this unit will reset
 */
void CPoultryUnit::Reset(){
    owner_ = -1;
    egg_number_ = 0;
}
```

```
/* ------------------------------------------------------------------
// COrchardUnit.h
// ------------------------------------------------------------------ */

#include"cmapunit.h"
#ifndef CORCHARDUNIT_H
#define CORCHARDUNIT_H
class COrchardUnit : public CMapUnit
{
    public:
        COrchardUnit() : state_(0), damage_level_(0){};
        COrchardUnit(const COrchardUnit &rhs);
        COrchardUnit & operator = (const COrchardUnit &rhs);
        virtual ~COrchardUnit();
        virtual void PlayerVisit(CPlayer &);
        virtual void Turn();
        virtual void Reset();
        const int GetState();
        const int GetDamege();
    private:
        int state_; /*0~4*/
        int damage_level_; /*0~4*/
};
#endif


/* ------------------------------------------------------------------
// COrchardUnit.cpp
// ------------------------------------------------------------------ */

#include"corchardunit.h"
#include<iostream>
using namespace std;

COrchardUnit::~COrchardUnit(){}

const int COrchardUnit::GetState()
{    return state_;     }
```

```
const int COrchardUnit::GetDamege()
{    return damage_level_;   }
/*
 * PlayerVisit : when player visit this unit
 */
void COrchardUnit::PlayerVisit(CPlayer &player)
{
    if(player.GetIdentifier() == owner_)
    {
        if(state_ != 4) /* can't harvest */
        {
            cout << "Do you want to SEEDING this Unit?(y/n) ";
            char choice;
            cin >> choice;
            if(choice == 'y')
            {
                player.AddMoney(-1 * seeding_price_);
                state_++;
                cout << "Spend " << seeding_price_ << " for SEEDING..... ";
            }

        }
        else /* can harvest */
        {
            int pass = rand() % (damage_level_ + 1);
            if(pass == 0) /* harvest success */
            {
                cout << "harvest, add money " << harvest_price_<< endl;
                player.AddMoney(harvest_price_);
            }
            else /* harvest failure */
            {
                cout << "harvest failure....." << endl;
            }
            damage_level_ = 0;
            state_ = 0;
        }
    }
```

```
    else if(owner_ == -1) /* no owned */
    {
        cout << "Do you want to buy this Unit?(y/n) ";
        char choice;
        cin >> choice;
        if(choice == 'y')
        {
            player.AddMoney(-1 * buy_price_);
            owner_ = player.GetIdentifier();
            player.SetOwned(player.GetOwned() + 1);
            cout << "Spend " << buy_price_ << " for BUY this Unit..... ";
        }
    }
    else /* others owned */ {
        int pass = rand() % 2;
        if(pass == 1) /* put bug success */{
            if(damage_level_ < 4 && state_ != 0)  {
                cout << "put bug...." << endl;
                damage_level_++;
            }
        }
        else /* put bug failure*/{
            cout << "nothing to happen...." << endl;
        }
    }
}
/*
 * Turn : every turn change on this unit
 */
void COrchardUnit::Turn(){}
/*
 * Reset : when owner is break, this unit will reset
 */
void COrchardUnit::Reset(){
    owner_ = 0;
    state_ = 0;
    damage_level_ = 0;
}
```

```
/* ----------------------------------------------------------------
// CLivestockUnit.h
// ---------------------------------------------------------------- */

#include <string>
using namespace std;
#include"cmapunit.h"

#ifndef CLIVESTOCKUNIT_H
#define CLIVESTOCKUNIT_H
class CLivestockUnit : public CMapUnit
{
    public:
        CLivestockUnit() : number_(0), turn_(0), live_period_(0), unit_(""){};
        CLivestockUnit(const CLivestockUnit &);
        CLivestockUnit & operator = (const CLivestockUnit &rhs);
        virtual ~CLivestockUnit();
        virtual void PlayerVisit(CPlayer &);
        virtual void Turn();
        virtual void Reset();
        void SetLivePeriod(const int live_period){ live_period_ = live_period; };
        void SetUnit(const string unit){ unit_ = unit; };
        const int GetNumber();
        const int GetTurn();
        const int GetLivePeriod(){ return live_period_; };
        const string GetUnit(){ return unit_; };
    private:
        int number_;
        int turn_;
        int live_period_;
        string unit_;
};
#endif
```

```
/* ------------------------------------------------------------------
// CLivestockUnit.cpp
// ------------------------------------------------------------------ */
#include"clivestockunit.h"
#include<iostream>
using namespace std;

CLivestockUnit::~CLivestockUnit(){}

const int CLivestockUnit::GetTurn(){   return turn_; }

const int CLivestockUnit::GetNumber()  return number_;}
/*
 * PlayerVisit : when player visit this unit
 */
void CLivestockUnit::PlayerVisit(CPlayer &player){
    if(player.GetIdentifier() == owner_){
        if(number_ == 0) /* can't harvest */   {
            cout << "SEEDING~~" << endl;
            player.AddMoney(-1 * seeding_price_);
        }
        else /* can harvest */  {
            cout << "harvest, add money " << harvest_price_ * number_ << endl;
            player.AddMoney(harvest_price_ * number_);
            number_ = 0;
        }
    }
    else if(owner_ == -1) /* no owned */   {
        cout << "Do you want to buy this Unit?(y/n) ";
        char choice;
        cin >> choice;
        if(choice == 'y')  {
            player.AddMoney(-1 * buy_price_);
            owner_ = player.GetIdentifier();
            player.SetOwned(player.GetOwned() + 1);
            cout << "Spend " << buy_price_ << " for BUY this Unit..... ";
        }
    }
```

```
    else /* others owned */
    {
        int pass = rand() % 2;
        if(pass == 1) /* steal success */
        {
            cout << "be stool.... decrease 1" << unit_ << endl;
            if(number_ > 0)
            {
                number_--;
            }
        }
        else /* steal failure*/      {
            cout << "nothing to happen...." << endl;
        }
    }
}

/*
 * Turn : every turn change on this unit
 */
void CLivestockUnit::Turn(){
    if(owner_ != -1)   {
        turn_ = ( turn_ + 1 ) % live_period_;
        if(turn_ == 0)     {
            number_++;
        }
    }

}
/*
 * Reset : when owner is break, this unit will reset
 */
void CLivestockUnit::Reset()
{
    owner_ = -1;
    number_ = 0;
    turn_ = 0;
}
```

```
/* ----------------------------------------------------------------
// CMapUnit.h
// ---------------------------------------------------------------- */

#include<string>
#include"cplayer.h"
#ifndef CMAPUNIT_H
#define CMAPUNIT_H
using namespace std;
class CMapUnit{
    public:
        CMapUnit() : buy_price_(0), seeding_price_(0), harvest_price_(0), owner_(-1),
mapname_(""){};
        CMapUnit(const CMapUnit &);
        CMapUnit & operator = (const CMapUnit &rhs);
        virtual ~CMapUnit(){};
        virtual void PlayerVisit(CPlayer &) = 0;
        virtual void Turn() = 0;
        virtual void Reset() = 0;
        void SetType(const char type){ type_ = type; };
        void SetBuyPrice(const int buy){ buy_price_ = buy; };
        void SetSeedingPrice(const int seed) { seeding_price_ = seed; };
        void SetHarvestPrice(const int harv){ harvest_price_ = harv; };
        void SetOwner(const int owner){ owner_ = owner; };
        void SetMapName(const string name){ mapname_ = name; };
        const int GetType(){ return type_; };
        const int GetBuyPrice(){ return buy_price_;};
        const int GetSeedingPrice(){ return seeding_price_;};
        const int GetHarvestPrice(){ return harvest_price_;};
        const int GetOwner(){ return owner_;};
        const string GetMapName(){ return mapname_; };
    protected:
        char type_;
        int buy_price_, seeding_price_, harvest_price_, owner_;
        string mapname_;

};
#endif
```

TPP09-HW4-496470516 高翊展

```
/* ------------------------------------------------------------------
// main.cpp
// ------------------------------------------------------------------ */

#include <cstdlib>
#include <iostream>
#include "cfarmgame.h"

using namespace std;

int main()
{
     CFarmGame game;
     game.Start();
    system("PAUSE");
    return 0;
}
```