struct Sklad // Объявили новую структуру склад

{

int ID\_Sklad;

int Kolichestvo\_Skladskih\_Mest; // количество складских мест

string Nathvanie\_Sklada; // название склада

Sklad \*next, \*Prev; // указатель на следующий элемент и предыдущий элемент

};

struct listSklad // задание списка ссылками

{

Sklad \*head; // Первый элемент (голова) списка

Sklad \*tail; // Последний элемент (хвост) списка

};

struct Komplektyushai // Объявили новую структуру комплекующая

{

int ID\_Komplektyushai;

string Naimenovanie; // Наименование

int Cena\_Komplektyushei; // Цена комплектующей

Komplektyushai \*next, \*Prev;

};

struct listKomplektyushai // задание списка ссылками

{

Sklad \*head; // Первый элемент (голова) списка

Sklad \*tail; // Последний элемент (хвост) списка

};

struct Master // Объявили новую структуру мастер

{

int ID\_Master;

string FIO; // ФИО

int Zarplata; // Зарплата

Master \*next, \*Prev;

};

struct listMaster // задание списка ссылками

{

Sklad \*head; // Первый элемент (голова) списка

Sklad \*tail; // Последний элемент (хвост) списка

};

struct Mesto\_Hranenia // Объявили новую структуру Место хранения

{

int ID\_Mesto\_Hranenia;

int ID\_Sklad;

int ID\_Komplektyushai;

int Kolichestvo\_Komplektyushih; // Количество комплектующих

Mesto\_Hranenia \*next, \*Prev;

};

struct listMesto\_Hranenia // задание списка ссылками

{

Sklad \*head; // Первый элемент (голова) списка

Sklad \*tail; // Последний элемент (хвост) списка

};

struct Remont // Объявили новую структуру Ремонт

{

int ID\_Remont;

int ID\_Komplektyushai;

int Stoimost; // Стоимость

int ID\_Master;

string Data\_Remonta; // Дата ремонта

Remont \*next, \*Prev;

};

struct listRemont // задание списка ссылками

{

Sklad \*head; // Первый элемент (голова) списка

Sklad \*tail; // Последний элемент (хвост) списка

};

struct Ispolzyemie\_Komplektyushie // Объявили новую структуру используемые комплектующие

{

int ID\_Ispolzyemie\_Komplektyushie;

int ID\_Komplektyushai;

int Kolichestvo;

int ID\_Remont;

Ispolzyemie\_Komplektyushie \*next, \*Prev;

};

struct listIspolzyemie\_Komplektyushie // задание списка ссылками

{

Sklad \*head; // Первый элемент (голова) списка

Sklad \*tail; // Последний элемент (хвост) списка

};

struct Vid\_Remonta

{

int ID\_Vid\_Remonta;

string Nazvanie; // Наименование

Vid\_Remonta \*next, \*Prev;

};

struct listVid\_Remonta

void List::addSklad(int ID\_s,int Kolichestvo\_Skladskih\_Mest,string Nathvanie\_Sklada)

{

for (Sklad \*temp = Sklad\_L->head; temp != NULL; temp = temp->next) { // Пока

if (temp->ID\_Sklad == ID\_s) { // если

cout << "Склад: ID = " << ID\_s << " уже существует" << endl;

return;

}

}

for (Sklad \*temp = Sklad\_L->head; temp != NULL; temp = temp->next)

{

if (temp->Nathvanie\_Sklada == Nathvanie\_Sklada)

{

cout << "Склад: Название склада = " << Nathvanie\_Sklada << " уже существует" << endl;

return;

}

}

Sklad \*temp = new Sklad;

temp->ID\_Sklad = ID\_s;

temp->Kolichestvo\_Skladskih\_Mest=Kolichestvo\_Skladskih\_Mest;

temp->Nathvanie\_Sklada = Nathvanie\_Sklada;

temp->next = Sklad\_L->head;

Sklad\_L->head = temp;

}

void List::addKomplektyushai (int ID\_K , string Naimenovanie, int Cena\_Komplektyushei)

{

for (Komplektyushai \*temp = Komplektyushai\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Komplektyushai == ID\_k)

{

cout << "Комплектующая: ID = " << ID\_k << " уже существует" << endl;

return;

}

}

for (Komplektyushai \*temp = Komplektyushai\_L->head; temp != NULL; temp = temp->next)

{

if (temp->Naimenovanie == Naimenovanie)

{

cout << "Комплектующая: Наименование = " << Naimenovanie << " уже существует" << endl;

return;

}

}

Komplektyushai \*temp = new Komplektyushai;

temp->ID\_Komplektyushai = ID\_k;

temp->Naimenovanie = Naimenovanie;

temp->Cena\_Komplektyushei = Cena\_Komplektyushei;

temp->next = Komplektyushai\_L->head;

Komplektyushai\_L->head = temp;

}

void List::addMaster (int ID\_m , string FIO, int Zarplata)

{

for (Master \*temp = Master\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Master == ID\_M)

{

cout << "Мастер: ID = " << ID\_M << " уже существует" << endl;

return;

}

}

for (Master \*temp = Master\_L->head; temp != NULL; temp = temp->next)

{

if (temp->FIO == FIO)

{

cout << "Мастер: ФИО = " << FIO << " уже существует" << endl;

return;

}

}

Master \*temp = new Master;

temp->ID\_Master = ID\_M;

temp->FIO = FIO;

temp->Zarplata = Zarplata;

temp->next = Master\_L->head;

Master\_L->head = temp;

}

void List::addMesto\_Hranenia (int ID\_Me , int ID\_S, int ID\_K, int Kolichestvo\_Komplektyushih)

{

for (Mesto\_Hranenia \*temp = Mesto\_Hranenia\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Mesto\_Hranenia == ID\_Me)

{

cout << "Место хранения: ID = " << ID\_Me << " уже существует" << endl;

return;

}

}

for (Sklad \*tmp = Sklad\_L->head; tmp != NULL; tmp = tmp->next)

{

for (Komplektyushai \*tmp2 = Komplektyushai\_L->head; tmp2 != NULL; tmp2 = tmp2->next)

{

if ((tmp->ID\_Sklad == ID\_S) && (tmp2->ID\_Komplektyushai == ID\_K))

{

Mesto\_Hranenia \*temp = new Mesto\_Hranenia;

temp->ID\_Mesto\_Hranenia = ID\_Me;

temp->ID\_Sklad = ID\_S;

temp->ID\_Komplektyushai = ID\_K;

temp->Kolichestvo\_Komplektyushih = Kolichestvo\_Komplektyushih;

temp->next = Mesto\_Hranenia->head;

Mesto\_Hranenia->head = temp;

return;

}

}

}

cout << "Этот \"ID склада \" и \"ID комплектующей\" не существуют" << endl;

}

void List::addVid\_Remonta(int ID\_V, string Nazvanie)

{

for (Vid\_Remonta \*temp = Vid\_Remonta\_Д->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Vid\_Remonta == ID\_V)

{

cout << "Вид ремонта: ID = " << ID\_V << " уже существует" << endl;

return;

}

}

for (Vid\_Remonta \*temp = Vid\_Remonta\_L->head; temp != NULL; temp = temp->next)

{

if (temp->Nazvanie == Nazvanie)

{

cout << "Вид ремонта: Название = " << Name << " уже существует" << endl;

return;

}

}

Vid\_Remonta \*temp = new Vid\_Remonta;

temp->ID\_Vid\_Remonta = ID\_V;

temp->Nazvanie = Nazvanie;

temp->next = Vid\_Remonta\_L->head;

Vid\_Remonta\_L->head = temp;

}

void List::addRemont (int ID\_R , int ID\_V,int Stoimost, int ID\_M, int Kolichestvo\_Komplektyushih,string Data\_Remonta)

{

for (Remont \*temp = Remont\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Remont == ID\_R)

{

cout << "Ремонт: ID = " << ID\_R << " уже существует" << endl;

return;

}

}

for (Vid\_Remonta \*tmp = Vid\_Remonta\_L->head; tmp != NULL; tmp = tmp->next)

{

for (Master \*tmp2 = Master\_L->head; tmp2 != NULL; tmp2 = tmp2->next)

{

if ((tmp->ID\_Vid\_Remonta == ID\_V) && (tmp2->ID\_Master == ID\_M))

{

Remont \*temp = new Remont;

temp->ID\_Remont = ID\_R;

temp->ID\_Vid\_Remonta = ID\_V;

temp->Stoimost = Stoimost;

temp->ID\_Master = ID\_M;

temp->Kolichestvo\_Komplektyushih = Kolichestvo\_Komplektyushih;

temp->Data\_Remonta = Data\_Remonta;

temp->next = Remont->head;

Remont->head = temp;

return;

}

}

}

cout << "Этот \"ID вид ремонта \" и \"ID мастера\" не существуют" << endl;

}

void List::addIspolzyemie\_Komplektyushie (int ID\_I , int ID\_K,int Kolichestvo, int ID\_R)

{

for (Ispolzyemie\_Komplektyushie \*temp = Ispolzyemie\_Komplektyushie\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Ispolzyemie\_Komplektyushie == ID\_I)

{

cout << "Используемые комплектующие: ID = " << ID\_I << " уже существует" << endl;

return;

}

}

for (Komplektyushai \*tmp = Komplektyushai\_L->head; tmp != NULL; tmp = tmp->next)

{

for (Remont \*tmp2 = Remont\_L->head; tmp2 != NULL; tmp2 = tmp2->next)

{

if ((tmp->ID\_Komplektyushai == ID\_K) && (tmp2->ID\_Remont == ID\_R))

{

Ispolzyemie\_Komplektyushie \*temp = new Ispolzyemie\_Komplektyushie;

temp->ID\_Ispolzyemie\_Komplektyushie = ID\_I;

temp->ID\_Komplektyushai = ID\_K;

temp->Kolichestvo = Kolichestvo;

temp->ID\_Remont = ID\_R;

temp->next = Ispolzyemie\_Komplektyushie->head;

Ispolzyemie\_Komplektyushie->head = temp;

return;

}

}

}

cout << "Этот \"ID комплектующая \" и \"ID ремонт\" не существуют" << endl;

}

void List::findByIDSklad(int ID\_S, int KolichestvoSkladskihMest, string NathvanieSklada)

{

for (Sklad \*temp = Sklad\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Sklad == ID\_S)

{

temp->Kolichestvo\_Skladskih\_Mest = KolichestvoSkladskihMest;

temp->Nathvanie\_Sklada = NathvanieSklada;

}

}

}

void List::findByIDKomplektyushai(int ID\_K, string Naim, int CenaKomplektyushei)

{

for (Komplektyushai \*temp = Komplektyushai\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Komplektyushai == ID\_K)

{

temp->Naimenovanie = Naim;

temp->Cena\_Komplektyushei = CenaKomplektyushei;

}

}

}

void List::findByIDMaster(int ID\_m, string F\_I\_O, int Zarplat)

{

for (Master \*temp = Master\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Master == ID\_m)

{

temp->FIO = F\_I\_O;

temp->Zarplata = Zarplat;

}

}

}

void List::findByIDMesto\_Hranenia(int ID\_me, int ID\_S, int ID\_K, int KolichestvoKomplektyushih)

{

for (Mesto\_Hranenia \*temp = Mesto\_Hranenia\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Mesto\_Hranenia == ID\_me)

{

temp->ID\_Sklad = ID\_S;

temp->ID\_Komplektyushai = ID\_K;

temp->Kolichestvo\_Komplektyushih = KolichestvoKomplektyushih;

}

}

}

void List::findByIDRemont(int ID\_R, int ID\_V, int Stoim, int ID\_M, string Data\_R)

{

for (Remont \*temp = Remont\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Remont == ID\_R)

{

temp->ID\_Vid\_Remonta = ID\_V;

temp->Stoimost = Stoim;

temp->ID\_Master = ID\_M;

temp->Data\_Remonta = Data\_R;

}

}

}

void List::findByIDIspolzyemie\_Komplektyushie(int ID\_I, int ID\_K, int Kol, int ID\_R)

{

for (Ispolzyemie\_Komplektyushie \*temp = Ispolzyemie\_Komplektyushie\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Ispolzyemie\_Komplektyushie == ID\_I)

{

temp->ID\_Komplektyushai = ID\_K;

temp->Kolichestvo = Kol;

temp->ID\_Remont = ID\_R;

}

}

}

void List::findByIDVid\_Remonta(int ID\_V, string Nazv)

{

for (Vid\_Remonta \*temp = Vid\_Remonta\_L->head; temp != NULL; temp = temp->next)

{

if (temp->ID\_Vid\_Remonta == ID\_V)

{

temp->ID\_Komplektyushai = ID\_K;

temp->Nazvanie = Nazv;

}

}

}