Постановка задачи

Задание (Вариант 4).Иерархия графических компонентов. Имеется набор компонентов: окно, надпись, список, кнопка, панель, линия. Компоненты окно и панель могут содержать другие компоненты. Компоненты могут быть видимы или скрыты, компоненты можно перемещать в пределах контейнера. Требуется отобразить иерархию компонентов и навигацию по ней в виде дерева.

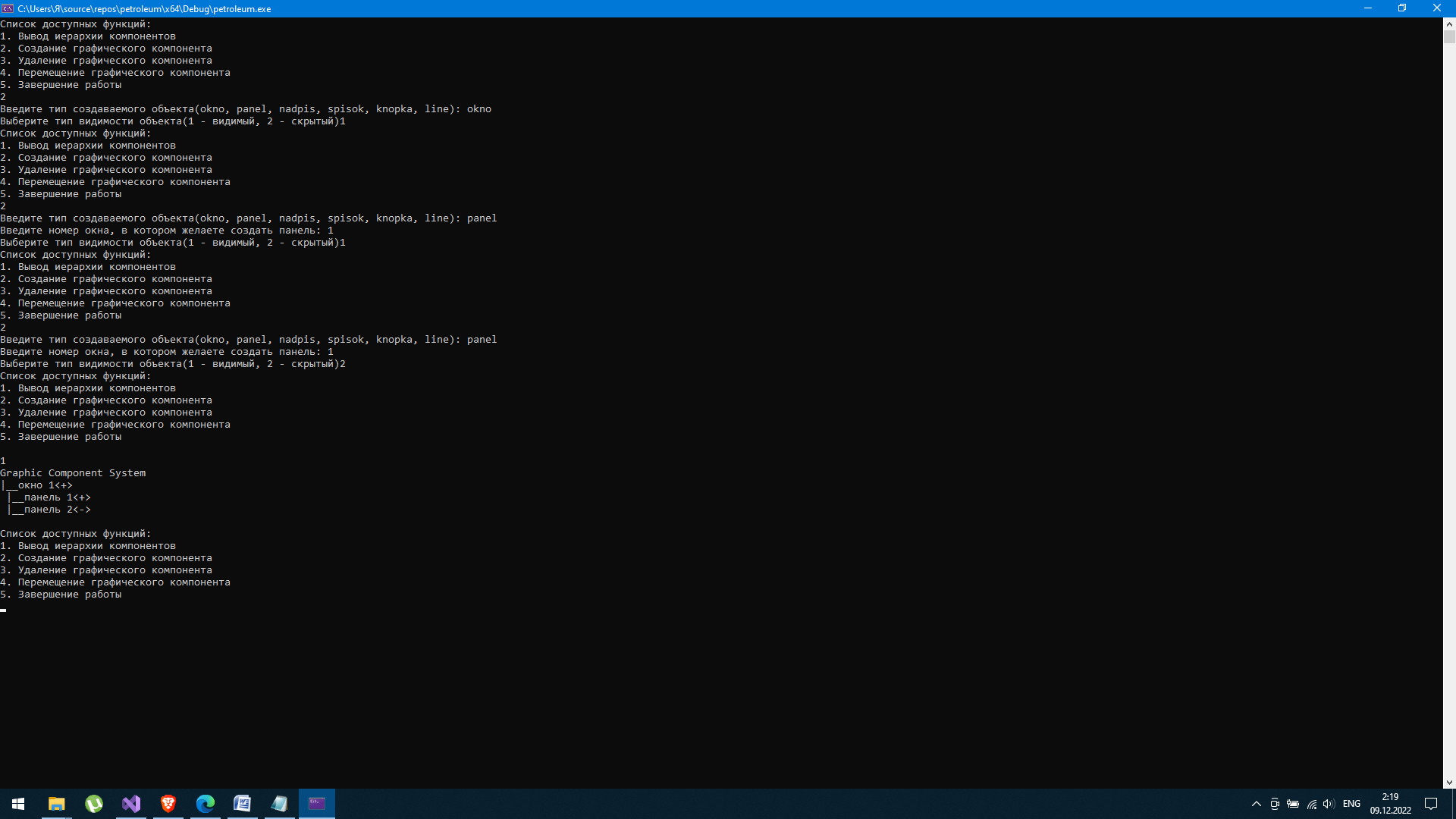


Рисунок 1 – Создание нескольких компонентов разного типа

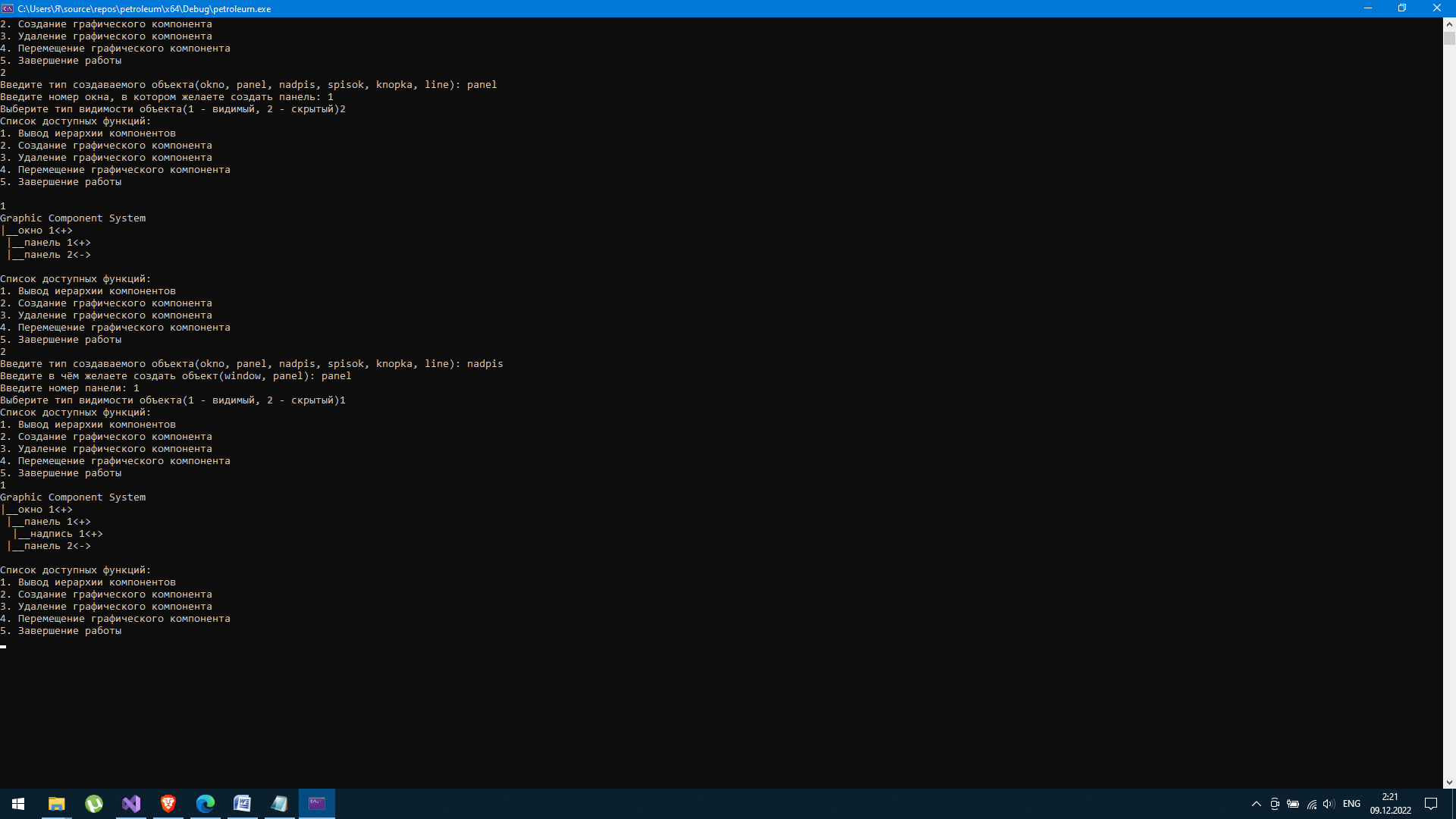


Рисунок 2 – Вывод иерархии после создания нескольких компонентов

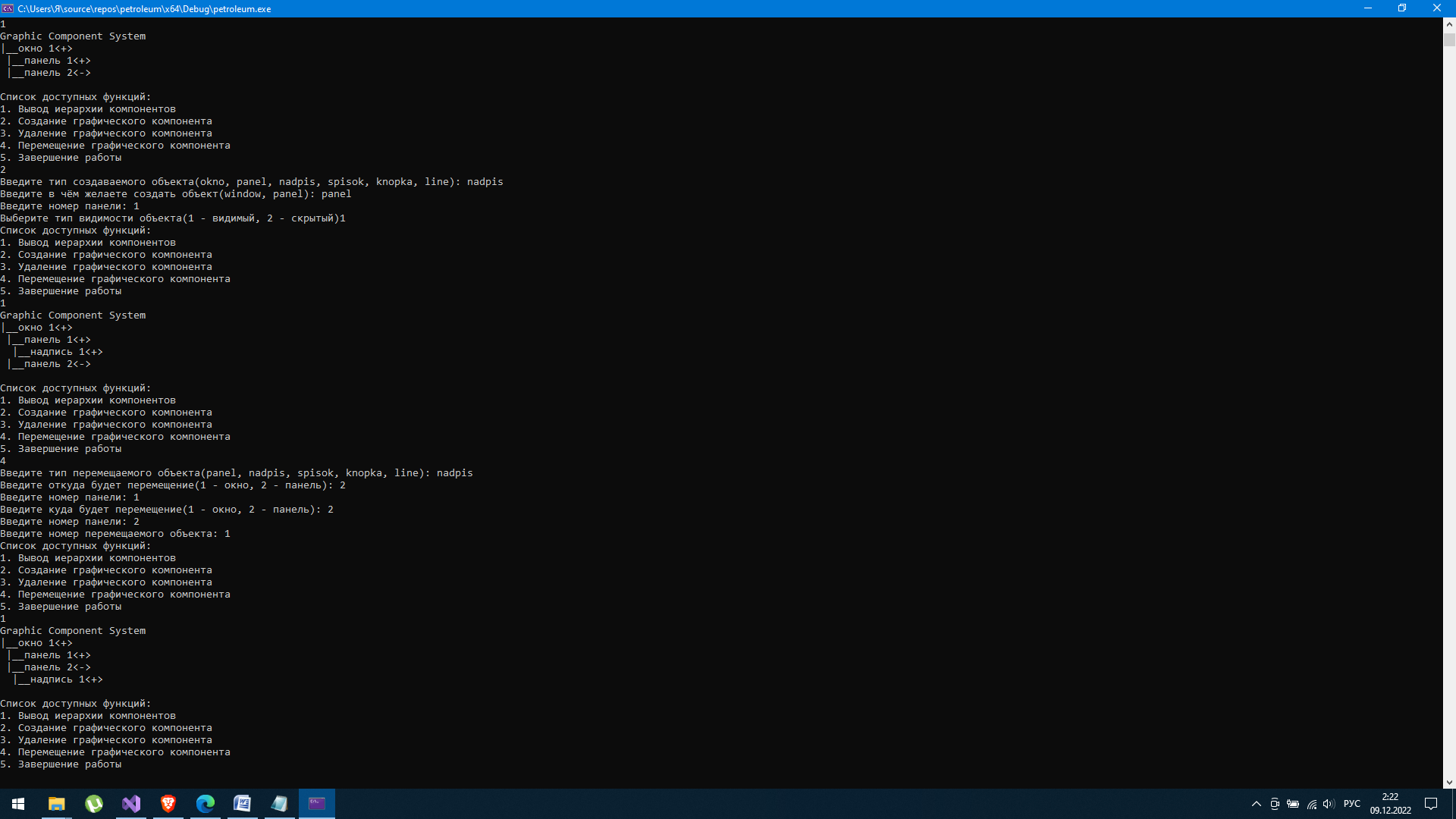


Рисунок 3 – Перемещение графического компонента надпись из первой панели во вторую

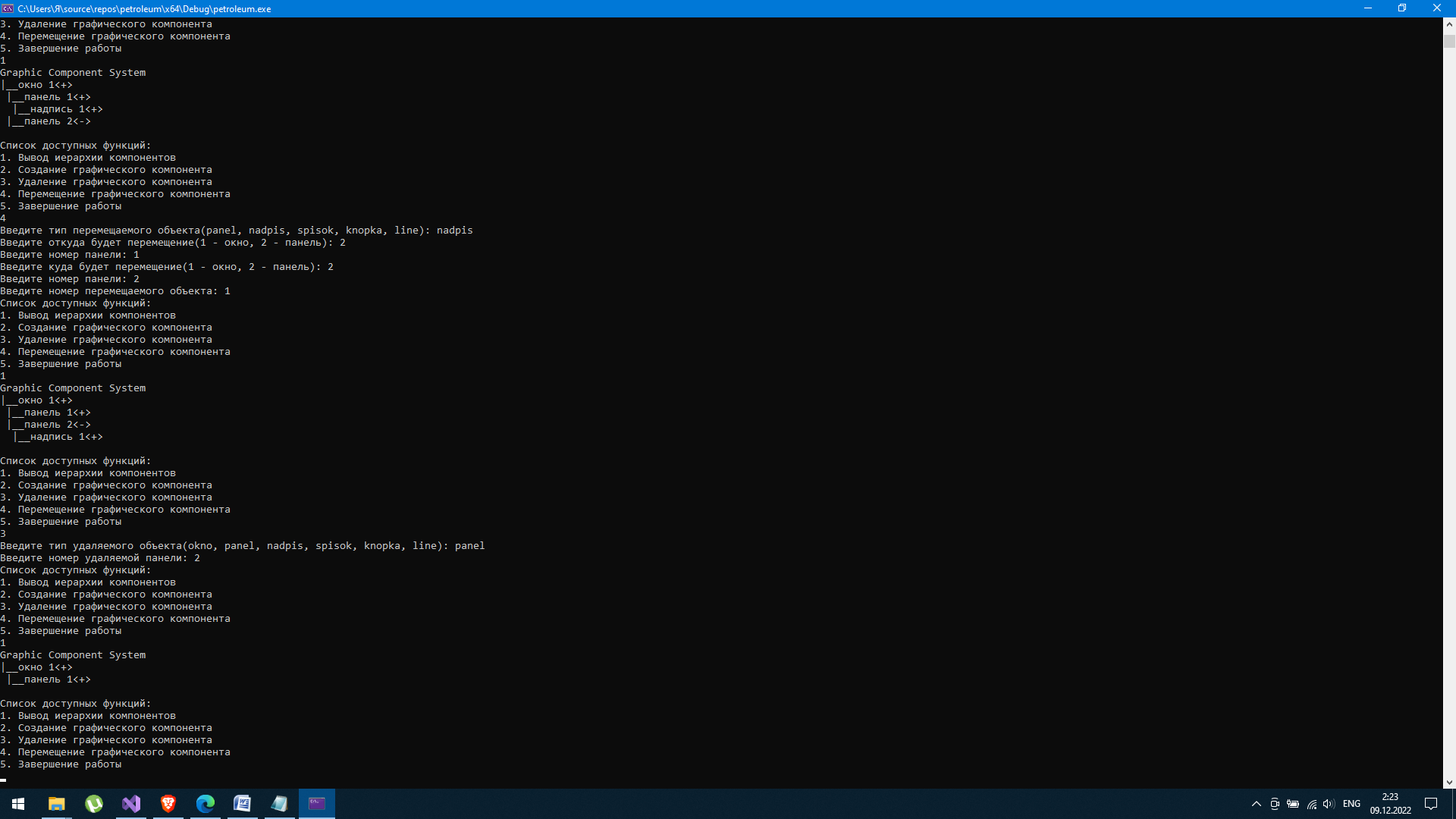


Рисунок 4 – Удаление второй панели с вложенной в неё надписью

Приложение

Полный текст программы:

#include <iostream>  
#include <string>  
using namespace std;  
class father\_object  
{  
protected:  
 string name;  
 string visibl;  
 int first\_numb;  
 int numb;  
 int second\_numb;  
 int kol\_ins;  
 int kol\_list;  
 int kol\_buttom;  
 int kol\_panel;  
 int kol\_line;  
public:  
 string get\_name() { return name; };  
 string get\_visibl() { return visibl; };  
 int get\_first\_numb() { return first\_numb; };  
 int get\_second\_numb() { return second\_numb; };  
 int get\_kol\_ins() { return kol\_ins; };  
 int get\_kol\_list() { return kol\_list; };  
 int get\_kol\_buttom() { return kol\_buttom; };  
 int get\_kol\_panel() { return kol\_panel; };  
 int get\_kol\_line() { return kol\_line; };  
 int get\_numb() { return numb; };  
 int up\_kol\_ins() { return (kol\_ins = kol\_ins + 1); };  
 int up\_kol\_list() { return (kol\_list = kol\_list + 1); };  
 int up\_kol\_buttom() { return (kol\_buttom = kol\_buttom + 1); };  
 int up\_kol\_panel() { return (kol\_panel = kol\_panel + 1); };  
 int up\_kol\_line() { return (kol\_line = kol\_line + 1); };  
 int down\_kol\_ins() { return (kol\_ins = kol\_ins - 1); };  
 int down\_kol\_list() { return (kol\_list = kol\_list - 1); };  
 int down\_kol\_buttom() { return (kol\_buttom = kol\_buttom - 1); };  
 int down\_kol\_panel() { return (kol\_panel = kol\_panel - 1); };  
 int down\_kol\_line() { return (kol\_line = kol\_line - 1); };  
};  
class okno : public father\_object// создаётся независимо от всех, остальные к нему делают привязку  
{  
public:  
 okno()  
 {  
 kol\_ins = 0;  
 kol\_list = 0;  
 kol\_buttom = 0;  
 kol\_panel = 0;  
 kol\_line = 0;  
 }  
 void create(okno\*& win, int& N\_win, int& numb\_obj, int& sys\_numb)  
 {  
 int new\_N = N\_win + 1;  
 okno\* new\_win = new okno[new\_N];  
 for (int i = 0; i < N\_win; i++)  
 {  
 new\_win[i] = win[i];  
 }  
 new\_win[N\_win].name = "окно " + to\_string(numb\_obj);  
 new\_win[N\_win].numb = numb\_obj;  
 new\_win[N\_win].second\_numb = sys\_numb;  
 cout << "Выберите тип видимости объекта(1 - видимый, 2 - скрытый)";  
 int visibl;  
 cin >> visibl;  
 if (visibl == 1)  
 new\_win[N\_win].visibl = "<+>";  
 else if (visibl == 2)  
 new\_win[N\_win].visibl = "<->";  
 delete[] win;  
 win = new\_win;  
 N\_win = new\_N;  
 }  
};  
class panel : public father\_object  
{  
public:  
 panel()  
 {  
 kol\_ins = 0;  
 kol\_list = 0;  
 kol\_buttom = 0;  
 kol\_line = 0;  
 }  
 void create(okno\*& win, panel\*& pan, int& N\_win, int& N\_pan, int& numb\_obj, int& sys\_numb)  
 {  
 string name\_father;  
 int sch;  
 int new\_N = N\_pan + 1;  
 panel\* new\_pan = new panel[new\_N];  
 for (int i = 0; i < N\_pan; i++)  
 {  
 new\_pan[i] = pan[i];  
 }  
 cout << "Введите номер окна, в котором желаете создать панель: ";  
 cin >> sch;  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (win[i].get\_numb() == sch)  
 {  
 new\_pan[N\_pan].first\_numb = win[i].get\_second\_numb();  
 new\_pan[N\_pan].name = "панель " + to\_string(numb\_obj);  
 new\_pan[N\_pan].numb = numb\_obj;  
 new\_pan[N\_pan].second\_numb = sys\_numb;  
 win[i].up\_kol\_panel();  
 break;  
 }  
 }  
 cout << "Выберите тип видимости объекта(1 - видимый, 2 - скрытый)";  
 int visibl;  
 cin >> visibl;  
 if (visibl == 1)  
 new\_pan[N\_pan].visibl = "<+>";  
 else if (visibl == 2)  
 new\_pan[N\_pan].visibl = "<->";  
 delete[] pan;  
 pan = new\_pan;  
 N\_pan = new\_N;  
 }  
 void move\_pan(okno\*& win, panel\*& pan, int& N\_win, int& N\_pan)  
 {  
 int sch\_old, sch\_new, old\_numb, new\_numb, numb\_obj, new\_ind\_obj, numb\_old, numb\_new;  
  
 cout << "Введите номер окна, откуда будет совершено перемещение: ";  
 cin >> numb\_old;  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (win[i].get\_numb() == numb\_old)  
 {  
 win[i].down\_kol\_panel();  
 break;  
 }  
 }  
 cout << "Введите номер окна, в которе будет совершено перемещение: ";  
 cin >> numb\_new;  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (win[i].get\_numb() == numb\_new)  
 {  
 new\_ind\_obj = win[i].get\_second\_numb();  
 win[i].up\_kol\_panel();  
 break;  
 }  
 }  
 cout << "Введите номер перемещаемого объекта: ";  
 cin >> numb\_obj;  
 for (int i = 0; i < N\_pan; i++)  
 {  
 if (pan[i].get\_numb() == numb\_obj)  
 {  
 pan[i].first\_numb = new\_ind\_obj;  
 break;  
 }  
 }  
 }  
};  
class object  
{  
protected:  
 string name;  
 int numb;  
 int first\_numb;  
 int second\_numb;  
 string visibl;  
public:  
 string get\_name() { return name; };  
 int get\_first\_numb() { return first\_numb; };  
 int get\_second\_numb() { return second\_numb; };  
 int get\_numb() { return numb; };  
 string get\_visibl() { return visibl; };  
 template <typename T>  
 void create(okno\*& win, panel\*& panel, T\*& obj, int& N\_win, int& N\_panel, int& N\_obj, int& system\_numb, int& numb\_obj, int type)  
 {  
 string name\_father;  
 string sch;  
 int numb;  
 int new\_N = N\_obj + 1;  
 cout << "Введите в чём желаете создать объект(window, panel): ";  
 getline(cin, sch);  
 T\* new\_obj = new T[new\_N];  
 for (int i = 0; i < N\_obj; i++)  
 {  
 new\_obj[i] = obj[i];  
 }  
 while (true)  
 {  
 if (sch == "window")  
 {  
 cout << "Введите номер окна: ";  
 cin >> numb;  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (win[i].get\_numb() == numb)  
 {  
 new\_obj[N\_obj].first\_numb = win[i].get\_second\_numb();  
 new\_obj[N\_obj].second\_numb = system\_numb;  
 if (type == 1)  
 new\_obj[N\_obj].name = "надпись " + to\_string(numb\_obj);  
 if (type == 2)  
 new\_obj[N\_obj].name = "список " + to\_string(numb\_obj);  
 if (type == 3)  
 new\_obj[N\_obj].name = "кнопка " + to\_string(numb\_obj);  
 if (type == 4)  
 new\_obj[N\_obj].name = "линия " + to\_string(numb\_obj);  
 new\_obj[N\_obj].numb = numb\_obj;  
 if (type == 1)  
 win[i].up\_kol\_ins();  
 if (type == 2)  
 win[i].up\_kol\_list();  
 if (type == 3)  
 win[i].up\_kol\_buttom();  
 if (type == 4)  
 win[i].up\_kol\_line();  
 break;  
 }  
 }  
 break;  
 }  
 else if (sch == "panel")  
 {  
 cout << "Введите номер панели: ";  
 cin >> numb;  
 for (int i = 0; i < N\_panel; i++)  
 {  
 if (panel[i].get\_numb() == numb)  
 {  
 new\_obj[N\_obj].first\_numb = panel[i].get\_second\_numb();  
 new\_obj[N\_obj].second\_numb = system\_numb;  
 new\_obj[N\_obj].name = "надпись " + to\_string(numb\_obj);  
 new\_obj[N\_obj].numb = numb\_obj;  
 if (type == 1)  
 panel[i].up\_kol\_ins();  
 if (type == 2)  
 panel[i].up\_kol\_list();  
 if (type == 3)  
 panel[i].up\_kol\_buttom();  
 if (type == 4)  
 panel[i].up\_kol\_line();  
 break;  
 }  
 }  
 break;  
 }  
 else  
 {  
 cout << "такого варианта нет!" << endl;  
 }  
 }  
 cout << "Выберите тип видимости объекта(1 - видимый, 2 - скрытый)";  
 int visibl;  
 cin >> visibl;  
 if (visibl == 1)  
 new\_obj[N\_obj].visibl = "<+>";  
 else if (visibl == 2)  
 new\_obj[N\_obj].visibl = "<->";  
 delete[] obj;  
 obj = new\_obj;  
 N\_obj = new\_N;  
 }  
 template <typename T>  
 void del(okno\*& win, panel\*& pan, T\*& obj, int& N\_win, int& N\_obj, int& N\_pan, int numb\_del, int type)  
 {  
 int new\_N = N\_obj - 1;  
 T\* new\_obj = new T[new\_N];  
 int numb\_delete;  
 for (int i = 0; i < N\_obj; i++)  
 {  
 if (obj[i].get\_numb() == numb\_del)  
 {  
 numb\_delete = i;  
 break;  
 }  
 }  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (obj[numb\_delete].get\_first\_numb() == win[i].get\_second\_numb())  
 {  
 if (type == 1)  
 win[i].down\_kol\_ins();  
 if (type == 2)  
 win[i].down\_kol\_list();  
 if (type == 3)  
 win[i].down\_kol\_buttom();  
 if (type == 4)  
 win[i].down\_kol\_line();  
 break;  
 }  
 }  
 for (int i = 0; i < N\_pan; i++)  
 {  
 if (obj[numb\_delete].get\_first\_numb() == pan[i].get\_second\_numb())  
 {  
 if (type == 1)  
 pan[i].down\_kol\_ins();  
 if (type == 2)  
 pan[i].down\_kol\_list();  
 if (type == 3)  
 pan[i].down\_kol\_buttom();  
 if (type == 4)  
 pan[i].down\_kol\_line();  
 break;  
 }  
 }  
 int j = 0;  
 for (int i = 0; i < N\_obj; i++)  
 {  
 if (i != numb\_delete)  
 {  
 new\_obj[j] = obj[i];  
 j++;  
 }  
 }  
 delete[] obj;  
 obj = new\_obj;  
 N\_obj = new\_N;  
 }  
 template <typename T>  
 void move(okno\*& win, panel\*& pan, T\*& obj, int& N\_win, int& N\_obj, int& N\_pan, int type)  
 {  
 int sch\_old, sch\_new, old\_numb, new\_numb, numb\_obj, new\_ind\_obj, numb\_old, numb\_new;  
 cout << "Введите откуда будет перемещение(1 - окно, 2 - панель): ";  
 cin >> sch\_old;  
 if (sch\_old == 1)  
 {  
 cout << "Введите номер окна: ";  
 cin >> numb\_old;  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (win[i].get\_numb() == numb\_old)  
 {  
 if (type == 1)  
 win[i].down\_kol\_ins();  
 if (type == 2)  
 win[i].down\_kol\_list();  
 if (type == 3)  
 win[i].down\_kol\_buttom();  
 if (type == 4)  
 win[i].down\_kol\_line();  
 break;  
 }  
 }  
 }  
 else if (sch\_old == 2)  
 {  
 cout << "Введите номер панели: ";  
 cin >> numb\_old;  
 for (int i = 0; i < N\_pan; i++)  
 {  
 if (pan[i].get\_numb() == numb\_old)  
 {  
 if (type == 1)  
 pan[i].down\_kol\_ins();  
 if (type == 2)  
 pan[i].down\_kol\_list();  
 if (type == 3)  
 pan[i].down\_kol\_buttom();  
 if (type == 4)  
 pan[i].down\_kol\_line();  
 break;  
 }  
 }  
 }  
 cout << "Введите куда будет перемещение(1 - окно, 2 - панель): ";  
 cin >> sch\_new;  
 if (sch\_new == 1)  
 {  
 cout << "Введите номер окна: ";  
 cin >> numb\_new;  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (win[i].get\_numb() == numb\_new)  
 {  
 new\_ind\_obj = win[i].get\_second\_numb();  
 if (type == 1)  
 win[i].up\_kol\_ins();  
 if (type == 2)  
 win[i].up\_kol\_list();  
 if (type == 3)  
 win[i].up\_kol\_buttom();  
 if (type == 4)  
 win[i].up\_kol\_line();  
 break;  
 }  
 }  
 }  
 else if (sch\_new == 2)  
 {  
 cout << "Введите номер панели: ";  
 cin >> numb\_new;  
 for (int i = 0; i < N\_pan; i++)  
 {  
 if (pan[i].get\_numb() == numb\_new)  
 {  
 new\_ind\_obj = pan[i].get\_second\_numb();  
 if (type == 1)  
 pan[i].up\_kol\_ins();  
 if (type == 2)  
 pan[i].up\_kol\_list();  
 if (type == 3)  
 pan[i].up\_kol\_buttom();  
 if (type == 4)  
 pan[i].up\_kol\_line();  
 break;  
 }  
 }  
 }  
 cout << "Введите номер перемещаемого объекта: ";  
 cin >> numb\_obj;  
 for (int i = 0; i < N\_obj; i++)  
 {  
 if (obj[i].get\_numb() == numb\_obj)  
 {  
 obj[i].first\_numb = new\_ind\_obj;  
 break;  
 }  
 }  
 }  
};  
class inscription : public object  
{  
};  
class list : public object  
{  
};  
class buttom : public object  
{  
};  
class line : public object  
{  
};  
void del\_panel(okno\*& win, panel\*& pan, inscription\*& ins, list\*& lis, buttom\*& but, line\*& lin, int& N\_win, int& N\_pan, int& N\_ins, int& N\_but, int& N\_lin, int N\_lis, int numb\_del);  
void del\_win(okno\*& win, panel\*& pan, inscription\*& ins, list\*& lis, buttom\*& but, line\*& lin, int& N\_win, int& N\_pan, int& N\_ins, int& N\_lis, int& N\_but, int& N\_lin, int numb\_del);  
void del\_win(okno\*& win, panel\*& pan, inscription\*& ins, list\*& lis, buttom\*& but, line\*& lin, int& N\_win, int& N\_pan, int& N\_ins, int& N\_lis, int& N\_but, int& N\_lin, int numb\_del)  
{  
 int delete\_numb;  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (win[i].get\_numb() == numb\_del)  
 {  
 delete\_numb = i;  
 break;  
 }  
 }  
 if (win[delete\_numb].get\_kol\_panel() > 0)  
 {  
 for (int i = 0; i < N\_pan; i++)  
 {  
 if (win[delete\_numb].get\_second\_numb() == pan[i].get\_first\_numb())  
 {  
 del\_panel(win, pan, ins, lis, but, lin, N\_win, N\_pan, N\_ins, N\_but, N\_lin, N\_lis, pan[i].get\_numb());  
 }  
 }  
 }  
 if (win[delete\_numb].get\_kol\_ins() > 0)  
 {  
 for (int i = 0; i < N\_ins; i++)  
 {  
 if (win[delete\_numb].get\_second\_numb() == ins[i].get\_first\_numb())  
 {  
 ins->del(win, pan, ins, N\_win, N\_ins, N\_pan, ins[i].get\_numb(), 1);  
 }  
 }  
 }  
 if (win[delete\_numb].get\_kol\_list() > 0)  
 {  
 for (int i = 0; i < N\_lis; i++)  
 {  
 if (win[delete\_numb].get\_second\_numb() == lis[i].get\_first\_numb())  
 {  
 lis->del(win, pan, lis, N\_win, N\_lis, N\_pan, lis[i].get\_numb(), 2);  
 }  
 }  
 }  
 if (win[delete\_numb].get\_kol\_buttom() > 0)  
 {  
 for (int i = 0; i < N\_but; i++)  
 {  
 if (win[delete\_numb].get\_second\_numb() == but[i].get\_first\_numb())  
 {  
 but->del(win, pan, but, N\_win, N\_but, N\_pan, but[i].get\_numb(), 3);  
 }  
 }  
 }  
 if (win[delete\_numb].get\_kol\_line() > 0)  
 {  
 for (int i = 0; i < N\_lin; i++)  
 {  
 if (win[delete\_numb].get\_second\_numb() == lin[i].get\_first\_numb())  
 {  
 lin->del(win, pan, lin, N\_win, N\_ins, N\_pan, lin[i].get\_numb(), 4);  
 }  
 }  
 }  
 int new\_N = N\_win - 1;  
 int j = 0;  
 okno\* new\_win = new okno[new\_N];  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (i != delete\_numb)  
 {  
 new\_win[j] = win[i];  
 j++;  
 }  
 }  
 delete[] win;  
 win = new\_win;  
 N\_win = new\_N;  
}  
void del\_panel(okno\*& win, panel\*& pan, inscription\*& ins, list\*& lis, buttom\*& but, line\*& lin, int& N\_win, int& N\_pan, int& N\_ins, int& N\_but, int& N\_lin, int N\_lis, int numb\_del)  
{  
 int delete\_numb;  
 for (int i = 0; i < N\_pan; i++)  
 {  
 if (pan[i].get\_numb() == numb\_del)  
 {  
 delete\_numb = i;  
 break;  
 }  
 }  
 if (pan[delete\_numb].get\_kol\_ins() > 0)  
 {  
 for (int i = 0; i < N\_ins; i++)  
 {  
 if (pan[delete\_numb].get\_second\_numb() == ins[i].get\_first\_numb())  
 {  
 ins->del(win, pan, ins, N\_win, N\_ins, N\_pan, ins[i].get\_numb(), 1);  
 }  
 }  
 }  
 if (pan[delete\_numb].get\_kol\_list() > 0)  
 {  
 for (int i = 0; i < N\_lis; i++)  
 {  
 if (pan[delete\_numb].get\_second\_numb() == lis[i].get\_first\_numb())  
 {  
 lis->del(win, pan, lis, N\_win, N\_lis, N\_pan, lis[i].get\_numb(), 2);  
 }  
 }  
 }  
 if (pan[delete\_numb].get\_kol\_buttom() > 0)  
 {  
 for (int i = 0; i < N\_but; i++)  
 {  
 if (pan[delete\_numb].get\_second\_numb() == but[i].get\_first\_numb())  
 {  
 but->del(win, pan, but, N\_win, N\_but, N\_pan, but[i].get\_numb(), 3);  
 }  
 }  
 }  
 if (pan[delete\_numb].get\_kol\_line() > 0)  
 {  
 for (int i = 0; i < N\_lin; i++)  
 {  
 if (pan[delete\_numb].get\_second\_numb() == lin[i].get\_first\_numb())  
 {  
 lin->del(win, pan, lin, N\_win, N\_ins, N\_pan, lin[i].get\_numb(), 4);  
 }  
 }  
 }  
 int new\_N = N\_pan - 1;  
 int j = 0;  
 for (int i = 0; i < N\_win; i++)  
 {  
 if (win[i].get\_second\_numb() == pan[delete\_numb].get\_first\_numb())  
 {  
 win[i].down\_kol\_panel();  
 break;  
 }  
 }  
 panel\* new\_pan = new panel[new\_N];  
 for (int i = 0; i < N\_pan; i++)  
 {  
 if (i != delete\_numb)  
 {  
 new\_pan[j] = pan[i];  
 j++;  
 }  
 }  
 delete[] pan;  
 pan = new\_pan;  
 N\_pan = new\_N;  
}  
void print\_panel(panel\*& pan, inscription\*& ins, list\*& lis, buttom\*& but, line\*& lin, int& N\_win, int& N\_pan, int& N\_ins, int& N\_but, int& N\_lin, int N\_lis, int kol\_prob, int tek\_numb)  
{  
  
 for (int i = 0; i < kol\_prob; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << pan[tek\_numb].get\_name() << pan[tek\_numb].get\_visibl() << endl;  
 if (pan[tek\_numb].get\_kol\_ins() > 0)  
 {  
 for (int i = 0; i < N\_ins; i++)  
 {  
 if (ins[i].get\_first\_numb() == pan[tek\_numb].get\_second\_numb())  
 {  
  
 for (int i = 0; i < kol\_prob + 1; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << ins[i].get\_name() << ins[i].get\_visibl() << endl;  
  
 }  
 }  
 }  
 if (pan[tek\_numb].get\_kol\_list() > 0)  
 {  
 for (int i = 0; i < N\_lis; i++)  
 {  
 if (lis[i].get\_first\_numb() == pan[tek\_numb].get\_second\_numb())  
 {  
  
 for (int i = 0; i < kol\_prob + 1; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << lis[i].get\_name() << lis[i].get\_visibl() << endl;  
  
 }  
 }  
 }  
 if (pan[tek\_numb].get\_kol\_buttom() > 0)  
 {  
 for (int i = 0; i < N\_but; i++)  
 {  
 if (but[i].get\_first\_numb() == pan[tek\_numb].get\_second\_numb())  
 {  
  
 for (int i = 0; i < kol\_prob + 1; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << but[i].get\_name() << but[i].get\_visibl() << endl;  
  
 }  
 }  
 }  
 if (pan[tek\_numb].get\_kol\_line() > 0)  
 {  
 for (int i = 0; i < N\_lin; i++)  
 {  
 if (lin[i].get\_first\_numb() == pan[tek\_numb].get\_second\_numb())  
 {  
  
 for (int i = 0; i < kol\_prob + 1; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << lin[i].get\_name() << lin[i].get\_visibl() << endl;  
  
 }  
 }  
 }  
  
}  
void print\_win(okno\*& win, panel\*& pan, inscription\*& ins, list\*& lis, buttom\*& but, line\*& lin, int& N\_win, int& N\_pan, int& N\_ins, int& N\_but, int& N\_lin, int N\_lis, int kol\_prob, int tek\_numb)  
{  
 for (int i = 0; i < kol\_prob; i++)  
 {  
 cout << " ";  
 }  
  
  
 cout << "|\_\_" << win[tek\_numb].get\_name() << win[tek\_numb].get\_visibl() << endl;  
 if (win[tek\_numb].get\_kol\_panel() > 0)  
 {  
 kol\_prob++;  
 for (int i = 0; i < N\_pan; i++)  
 {  
 if (pan[i].get\_first\_numb() == win[tek\_numb].get\_second\_numb())  
 {  
  
 print\_panel(pan, ins, lis, but, lin, N\_win, N\_pan, N\_ins, N\_but, N\_lin, N\_lis, kol\_prob, i);  
  
 }  
 }  
 kol\_prob--;  
 }  
 if (win[tek\_numb].get\_kol\_ins() > 0)  
 {  
 for (int i = 0; i < N\_ins; i++)  
 {  
 if (ins[i].get\_first\_numb() == win[tek\_numb].get\_second\_numb())  
 {  
  
 for (int i = 0; i < kol\_prob + 1; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << ins[i].get\_name() << ins[i].get\_visibl() << endl;  
  
 }  
 }  
 }  
 if (win[tek\_numb].get\_kol\_list() > 0)  
 {  
 for (int i = 0; i < N\_lis; i++)  
 {  
 if (lis[i].get\_first\_numb() == win[tek\_numb].get\_second\_numb())  
 {  
  
 for (int i = 0; i < kol\_prob + 1; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << lis[i].get\_name() << lis[i].get\_visibl() << endl;  
  
 }  
 }  
 }  
 if (win[tek\_numb].get\_kol\_buttom() > 0)  
 {  
 for (int i = 0; i < N\_but; i++)  
 {  
 if (but[i].get\_first\_numb() == win[tek\_numb].get\_second\_numb())  
 {  
  
 for (int i = 0; i < kol\_prob + 1; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << but[i].get\_name() << but[i].get\_visibl() << endl;  
  
 }  
 }  
 }  
 if (win[tek\_numb].get\_kol\_line() > 0)  
 {  
 for (int i = 0; i < N\_lin; i++)  
 {  
 if (lin[i].get\_first\_numb() == win[tek\_numb].get\_second\_numb())  
 {  
  
 for (int i = 0; i < kol\_prob + 1; i++)  
 {  
 cout << " ";  
 }  
 cout << "|\_\_" << lin[i].get\_name() << lin[i].get\_visibl() << endl;  
  
 }  
 }  
 }  
 //kol\_prob--;  
  
}  
int main()  
{  
 setlocale(0, "rus");  
 int sch, numb\_del;  
 string type\_object, name\_del;  
 int kol\_prob = 0, ind\_numb\_object = 0;  
 int N\_win = 0, N\_ins = 0, N\_list = 0, N\_but = 0, N\_pan = 0, N\_line = 0, numb\_win = 1, numb\_ins = 1, numb\_list = 1, numb\_but = 1, numb\_pan = 1, numb\_line = 1;  
 okno\* win = new okno[N\_win];  
 panel\* pan = new panel[N\_pan];  
 inscription\* ins = new inscription[N\_ins];  
 list\* lis = new list[N\_list];  
 buttom\* but = new buttom[N\_but];  
 line\* lin = new line[N\_line];  
 while (true)  
 {  
 cout << "Список доступных функций: " << endl;  
 cout << "1. Вывод иерархии компонентов" << endl;  
 cout << "2. Создание графического компонента" << endl;  
 cout << "3. Удаление графического компонента" << endl;  
 cout << "4. Перемещение графического компонента" << endl;  
 cout << "5. Завершение работы" << endl;  
 cin >> sch;  
 switch (sch)  
 {  
 case 1:  
 cout << "Graphic Component System" << endl;  
 for (int i = 0; i < N\_win; i++)  
 {  
 print\_win(win, pan, ins, lis, but, lin, N\_win, N\_pan, N\_ins, N\_but, N\_line, N\_list, kol\_prob, i);  
 }  
 cout << endl;  
 break;  
 case 2:  
 cout << "Введите тип создаваемого объекта(okno, panel, nadpis, spisok, knopka, line): ";  
 getline(cin, type\_object);  
 getline(cin, type\_object);  
 if (type\_object == "okno")  
 {  
 win->create(win, N\_win, numb\_win, ind\_numb\_object);  
 numb\_win++;  
 ind\_numb\_object++;  
 }  
 else if (type\_object == "panel")  
 {  
 pan->create(win, pan, N\_win, N\_pan, numb\_pan, ind\_numb\_object);  
 numb\_pan++;  
 ind\_numb\_object++;  
 }  
 else if (type\_object == "nadpis")  
 {  
 ins->create(win, pan, ins, N\_win, N\_pan, N\_ins, ind\_numb\_object, numb\_ins, 1);  
 ind\_numb\_object++;  
 numb\_ins++;  
 }  
 else if (type\_object == "spisok")  
 {  
 lis->create(win, pan, lis, N\_win, N\_pan, N\_list, ind\_numb\_object, numb\_list, 2);  
 ind\_numb\_object++;  
 numb\_list++;  
 }  
 else if (type\_object == "knopka")  
 {  
 but->create(win, pan, but, N\_win, N\_pan, N\_but, ind\_numb\_object, numb\_but, 3);  
 ind\_numb\_object++;  
 numb\_but++;  
 }  
 else if (type\_object == "line")  
 {  
 lin->create(win, pan, lin, N\_win, N\_pan, N\_line, ind\_numb\_object, numb\_list, 4);  
 ind\_numb\_object++;  
 numb\_line++;  
 }  
 else  
 {  
 cout << "Ошибка!" << endl;  
 }  
 break;  
 case 3:  
 cout << "Введите тип удаляемого объекта(okno, panel, nadpis, spisok, knopka, line): ";  
 getline(cin, type\_object);  
 getline(cin, type\_object);  
 if (type\_object == "okno")  
 {  
 cout << "Введите номер удаляемого окна: ";  
 cin >> numb\_del;  
 del\_win(win, pan, ins, lis, but, lin, N\_win, N\_pan, N\_ins, N\_list, N\_but, N\_line, numb\_del);  
 }  
 else if (type\_object == "panel")  
 {  
 cout << "Введите номер удаляемой панели: ";  
 cin >> numb\_del;  
 del\_panel(win, pan, ins, lis, but, lin, N\_win, N\_pan, N\_ins, N\_but, N\_line, N\_list, numb\_del);  
 }  
 else if (type\_object == "nadpis")  
 {  
 cout << "Введите номер удаляемой надписи: ";  
 cin >> numb\_del;  
 ins->del(win, pan, ins, N\_win, N\_ins, N\_pan, numb\_del, 1);  
 }  
 else if (type\_object == "spisok")  
 {  
 cout << "Введите номер удаляемого списка: ";  
 cin >> numb\_del;  
 lis->del(win, pan, lis, N\_win, N\_list, N\_pan, numb\_del, 2);  
 }  
 else if (type\_object == "knopka")  
 {  
 cout << "Введите номер удаляемой кнопки: ";  
 cin >> numb\_del;  
 but->del(win, pan, but, N\_win, N\_but, N\_pan, numb\_del, 3);  
 }  
 else if (type\_object == "line")  
 {  
 cout << "Введите номер удаляемой линии: ";  
 cin >> numb\_del;  
 lin->del(win, pan, lin, N\_win, N\_line, N\_pan, numb\_del, 4);  
 }  
 else  
 {  
 cout << "Ошибка! Объекта такого типа нет!" << endl;  
 }  
 break;  
 case 4:  
 cout << "Введите тип перемещаемого объекта(panel, nadpis, spisok, knopka, line): ";  
 getline(cin, type\_object);  
 getline(cin, type\_object);  
 if (type\_object == "panel")  
 {  
 pan->move\_pan(win, pan, N\_win, N\_pan);  
 }  
 else if (type\_object == "nadpis")  
 {  
 ins->move(win, pan, ins, N\_win, N\_ins, N\_pan, 1);  
 }  
 else if (type\_object == "spisok")  
 {  
 lis->move(win, pan, lis, N\_win, N\_list, N\_pan, 2);  
 }  
 else if (type\_object == "knopka")  
 {  
 but->move(win, pan, but, N\_win, N\_but, N\_pan, 3);  
 }  
 else if (type\_object == "line")  
 {  
 lin->move(win, pan, lin, N\_win, N\_line, N\_pan, 4);  
 }  
 else  
 {  
 cout << "Ошибка! Объекта такого типа нет!" << endl;  
 }  
 break;  
 case 5:  
 cout << "Благодарим вас за использование нашей программы!" << endl;  
 return 0;  
 break;  
 default:  
 cout << "Введённой вами команды в списке нет, выберите команду из списка существующих!" << endl;  
 break;  
  
 }  
 }  
 system("pause");  
 return 0;  
}