Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине «Проектирование программ в интеллектуальных системах»

Тема: «Типовой каркас оконного windows-приложения (ТКП). Обработка сообщений. Организация вывода в клиентскую область окна»

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Корнейчук А.И.

**Проверил:**

Муравьев Г.Л.

Брест 2022

**ЧАСТЬ 1**

**Цель работы:** 1. Ознакомиться со структурой и особенностями программирования оконных windows-приложений. 2. Изучить каркасы windows-приложений. 3. Изучить структуру типового каркаса оконных приложений (ТКП), использование ТКП для создания пользовательских приложений.

2. Создать каркас windows-приложения (тип Empty или аналогчный) средствами мастера (например, Win32 Application). Изучить его свойства (состав интерфейса; файловый состав; функциональный состав).

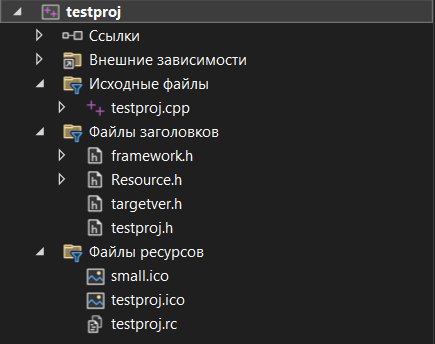
После создания пустого каркаса мы видим 3 папки в которых ничего нет, т.к. мы создали пустой каркас, так же при попытке запуска приложения вылезает ошибка, которая говорит нам о том что проект пустой и запускать нечего.

3. Создать проект из одного файла с одной, главной функцией, выводящей сообщения с помощью функций MessageBox. Для этого доработать каркас - создать файл в папке Source Files проекта приложения: - выбрать пункт меню File-New, в появившемся диалоговом окне выбрать вкладку файлы Files и пункт С++ Source File, т.е. создание файла с текстом на языке С++; - набрать имя файла в поле File name и завершить его создание; - включить в созданный файл заголовок #include <windows.h>, создать пустую функцию WinMain, например, как показано ниже и выполнить приложение

Функция MessageBox позволяет выводить разные шаблоны диалоговых окон, «привязывающиеся» к дескриптору hwnd или NULL.

4.Создать проект из двух файлов – главный (с расширением cpp) с текстом программы - функцией WinMain, заголовочный (с расширением h) с командами препроцессора. Для этого повторить п.3, модифицировать приложение, заголовочный файл подключить к главному командой препроцессора #include "\*.h". Выполнить приложение.

При реализации графических окон, остается возможность разделения на \*.h и \*.cpp файлы.

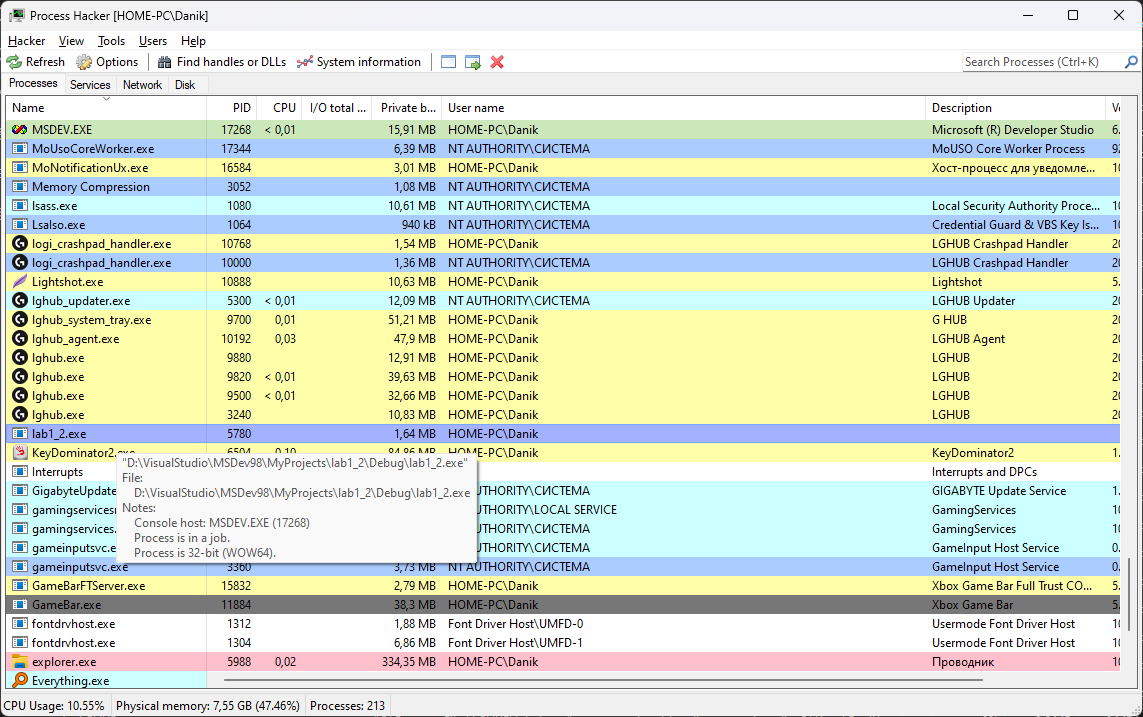
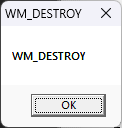
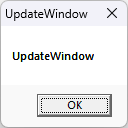
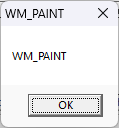
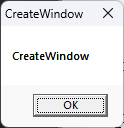
5. Создать каркас Windows-приложения (тип Simple или аналогичный) средствами мастера (аналогично п.2). Изучить его свойства: интерфейс; - файловый состав; - функциональный состав; - ресурсный состав.

В целом, функциональный состав определяет, что делает программа, ресурсный состав - как она это делает, а интерфейсный состав - как пользователь взаимодействует с программой и как программа взаимодействует с другими системами.

7. Изучить теоретический материал о сообщениях и типовом каркасе приложения. Cоздать приложение на базе типового каркаса приложения ТКП, взяв за основу автоматический каркас типа Simple (или аналогичный). Изучить его свойства: - интерфейс; - файловый состав; - функциональный состав; - ресурсный состав.

Файловый состав аналогичен каркасу типа Simple. Функциональный состав: ГЛАВНАЯ ФУНКЦИЯ int WINAPI WinMain и ФУНКЦИЯ-ОБРАБОТЧИК ГЛАВНОГО ОКНА LRESULT CALLBACK WndProc. Ресурсный состав: переменная глобального типа char szWindowStyle [ ] = "myWindowStyle"; Переменные в функции WinMain: HWND hWnd; MSG lpMsg ; WNDCLASS wcApp; Переменные в функции WndProc: HDC hdc; PAINTSTRUCT ps;

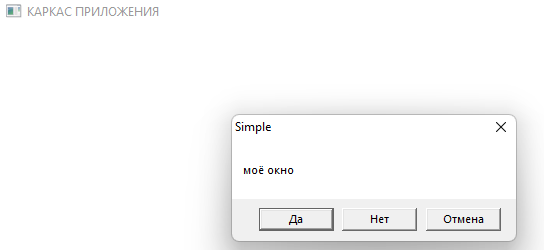
8. Модифицировать приложение, последовательно размещая MessageBox с сообщением о выполненной функции: а) после создания окна в CreateWindow; б) после вывода и после обновления окна UpdateWindow (hWnd); в) сразу в секции case WM\_DESTROY; г) в секции case WM\_PAINT в место для вставки фрагмента пользователя. Для каждого варианта вставки осуществить запуск приложения и рассмотреть его работу (вывод сообщений) при свертывании-развертывании окна, перекрытии, перемещении окна, попытке изменения размеров окна. Закомментировать case WM\_DESTROY: PostQuitMessage(0); . Выполнить несколько запусков ТКП и убедиться, что после закрытия приложений – они по-прежнему активны – но без окон – присутствуют в списке процессов компьютера!!!



При закомментировании процесса WM\_DESTROY, приложение пропадает из рабочего стола, но остаётся в рабочих процессах компьютера. Закрытие программы можно осуществить с помощью различных возможных программ, к примеру через диспетчер задач.

9. Повторить создание приложения (п. 7). Для этого: - создать каркас проекта, тип Simple (см. п.5); - в основной файл проекта (\*.cpp) вместо имеющегося там текста вставить текст ТКП (см. приложение 7 [1]), сохранив только команду #include "stdafx.h"; - выполнить приложение и при необходимости устранить ошибки, вызванные присутствием в тексте недоопределенных элементов. Вставить функции MessageBox.

При попытке запуска программы без исправлений, нам выдаёт ошибку о неопределённости переменных или методов, что и ожидалось увидеть, после исправления ошибок наше ТКП отработало должным образом. После вставки функции MessageBox, поверх нашего каркаса появляется диалоговое окно с сообщением для пользователя.



10. Cоздать приложение, модифицировав приложение на базе ТКП. Изменить его интерфейс (вид окна, состав элементов окна - кнопок и т.п.). Для этого: - создать приложение на базе ТКП - выполнить отмену опции окна - WS\_OVERLAPPEDWINDOW и обеспечить тот же состав окна с применением опции окна - WS\_OVERLAPPED путем поэлементного включения соответствующих элементов (кнопок и т.п.) через оператор “или” (см. приложение 5 [1]); hWnd = CreateWindow (szWindowStyle, "Karkas ", WS\_OVERLAPPED | WS\_CAPTION | WS\_SYSMENU | WS\_THICKFRAME | WS\_MINIMIZEBOX | WS\_MAXIMIZEBOX,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

(HWND) NULL,

(HMENU) NULL,

hInst NULL );

- выполнить отмену опции окна ТКП - WS\_OVERLAPPED и обеспечить тот же состав окна, но путем поэлементного включения соответствующих элементов (кнопок и т.п.).

hWnd = CreateWindow (szWindowStyle, "Karkas ", WS\_TILED | WS\_CAPTION | WS\_SYSMENU | WS\_THICKFRAME | WS\_MINIMIZEBOX | WS\_MAXIMIZEBOX,

CW\_USEDEFAULT,

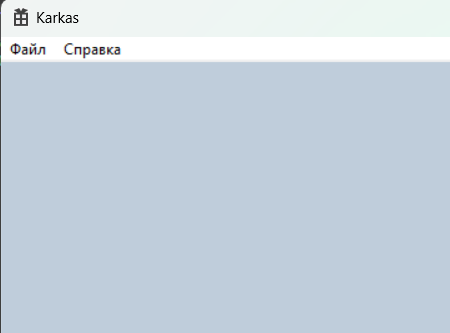
CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

(HWND) NULL,

(HMENU) NULL,

hInst NULL );

Внести изменения в параметры: - WNDCLASS.

- функции CreateWindow .Например, добавить в окно рамки, полосы прокрутки, убрать или добавить командные кнопки, внести изменения в координаты и размеры окна, использовав соответствующие параметры функции (x, y, Width, Height);

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_TESTPROJ2));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW-1);

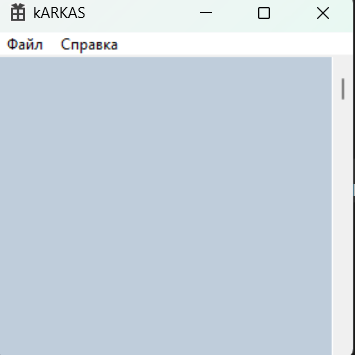
wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_TESTPROJ2);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

-функции ShowWindow .

ShowWindow (hWnd,SW\_SHOWMINIMIZED);



В WinAPI существует множество способов настроить окно(цвет, размер, функционал).Также есть возможность блокировать оконные-кнопки(типа “спрятано”, “показано”; “свернуто”, “развернуто”), вертикальные и горизонтальные полосы подкрутки, курсора, иконки и т.п.

11. Обеспечить попеременное переключение окна – “спрятано”, “показано”, “свернуто”, “развернуто”.

hWnd = CreateWindow (szWindowStyle, "kARKAS", WS\_BORDER | WS\_CAPTION | WS\_SYSMENU | WS\_VSCROLL|WS\_MAXIMIZEBOX|WS\_MINIMIZEBOX,100,200,300,

300, (HWND) NULL, (HMENU) NULL,hInst,NULL);

ShowWindow(hWnd, SW\_RESTORE);

Sleep(3);

ShowWindow(hWnd, SW\_MINIMIZE);

Sleep(3);

ShowWindow(hWnd, SW\_HIDE);

Sleep(3);

ShowWindow(hWnd, SW\_SHOW);

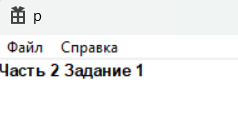
Константы WS\_MAXIMIZEBOX и WS\_MINIMIZEBOX как раз и отвечают за дынные переключения окон.

**Часть 2:**

**Цель работы:** 1. Ознакомиться с особенностями организации вывода в оконных windows-приложениях. 2. Ознакомиться с особенностями управления сообщениями.

**Ход работы:**

1. Cоздать приложение на базе ТКП. Добавить строку приветствия (координаты вывода - 0,0), для чего вставить в обработчик WM\_PAINT пользовательский фрагмент.



После вставки пользовательского фрагмента мы можем видеть в нашем ТКП сообщение «часть 2 задание 1»

Вывести в столбик для значений Х от 1 до 10 (с шагом 0,5) пары значений: Х - квадрат Х. Запустить приложение. Выполнить свертывание-развертывание окна, перемещение, изменение размеров окна. Убедиться в автоматической перерисовке окна.

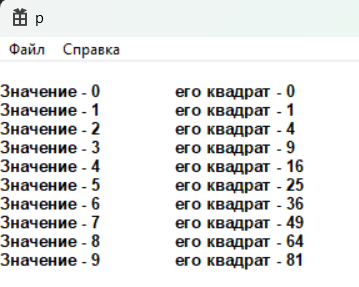
for (i=0; i<10; i++){

sprintf(MyString,"Значение - %d ", i);

TextOut ( hdc, 0, 0 + 15 \* (i + 1), MyString,strlen(MyString));

sprintf(MyString,"его квадрат - %d", i \* i);

TextOut ( hdc, 140, 0 + 15 \* (i + 1), MyString,strlen(MyString)); }



После проведения различных манипуляций с приложением, а именно свертывание, перемещение, изменение окна, мы наблюдали автоматическую перерисовку окна, которая никак не смещала и не изменяла наши данные которые мы поместили с помощью TextOut. Функция TextOut отвечает за вывод сообщения на заданном окне приложения(по дескриптору hdc, который и будет ссылаться на hWnd). При вызове на тех же координатах сообщение перерисовывается, поэтому при множестве TextOut следует использовать цикл и отделять с помощью итератора;

2.Модифицировать предыдущее приложение. Для этого: - изменить формат вывода и увеличить число обрабатываемых чисел до 75; - добавить вывод номера перерисовки (переменная int ReDrawNumber = 0).

++ReDrawNumber;

for (i=0; i<75; i++){

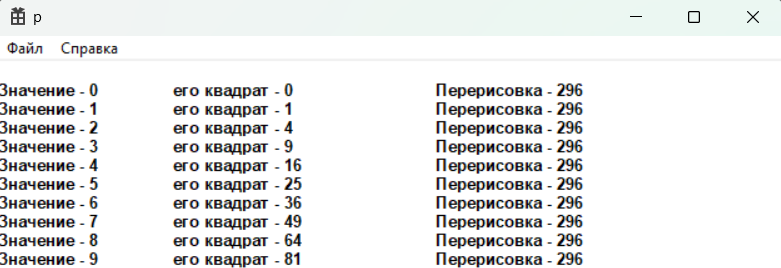
sprintf(MyString,"Значение- %d ", i);

TextOut ( hdc, 0, 0 + 15 \* (i + 1), MyString,strlen(MyString));

sprintf(MyString,"его квадрат - %d, перерисовка - %d", i \* i,ReDrawNumber);

TextOut ( hdc, 140, 0 + 15 \* (i + 1), MyString,strlen(MyString));}

Для того чтобы подсчёт перерисовки работал постоянно без обнуления счётчика перерисовок, переменная должна быть глобальной, то есть объявление переменной должно находиться вне функции, а само увеличение перерисовок внутри case WM\_PAINT:



Перерисовка считается изменение режимов окна, «свернуть»-«развернуть», «спрятать»-«показать», вывод за рабочий стол и возврат.

3. Cоздать приложение на базе ТКП. Организовать вывод графических данных. Основные графические примитивы рисуются с использованием библиотечных функций: - дуги эллипса Arc, ArcTo; - эллипсы и окружности Ellipse; - линии от текущей точки до точки, указанной в функции LineTo(hdc, x, y); - прямоугольники Rectangle; - полигоны Polygon; - связанные отрезки прямых PolyLine.

case WM\_PAINT:

hdc = BeginPaint( hWnd, &ps );

hPen = CreatePen(PS\_SOLID,1,RGB (122, 80, 219) );

SelectObject (hdc, hPen);

Ellipse ( hdc, 200, 80, 80, 360);

MoveToEx(hdc, 40, 200, NULL);

LineTo(hdc, 300, 300);

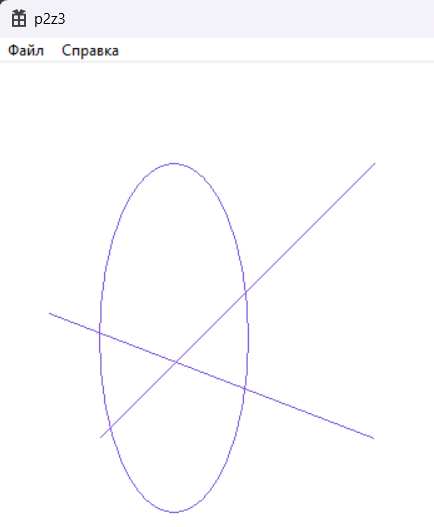
MoveToEx(hdc, 300, 80, NULL);

LineTo(hdc, 80,300);

ValidateRect(hWnd,NULL);

EndPaint(hWnd, &ps);

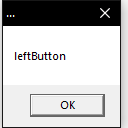
Существует нескольку функция для примитивного рисования фигур, например Ellipse, LineTo, Rectabgle и т.д. Функция CreatePen отвечает за “кисть”, которой программа будет отрисовывать эти фигуры.

4. Cоздать приложение на базе ТКП. Включить чувствительность к нажатию левой клавиши “мыши” (сообщение WM\_LBUTTONDOWN) - выводить окно MessageBox. Для этого создать в функции-обработчике дополнительную секцию.

case WM\_LBUTTONDOWN:

MessageBox(NULL,"leftButton","...",MB\_OK);

break;



WM\_LBUTTONDOWN отвечает за нажатие левой кнопки мыши.

Повторить п.п. 3-4, выполняя все действия в секции case WM\_LBUTTONDOWN (секция case WM\_PAINT должна быть пустой). Запустить приложение. Выполнить свертывание-развертывание окна, перемещение, перекрытие, изменение размеров окна. Проанализировать характер перерисовки содержимого окна.

case WM\_LBUTTONDOWN:

{

srand(NULL);

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

TextOut(hdc, rand()%250, rand()%250, L"temporary event", 20);

ValidateRect(hWnd, NULL);

EndPaint(hWnd, &ps);

}

break;

Свёртывание-развёртывание, перемещение, перекрытие, изменение размеров окна приводит к исчезновении надписи, но при повторном нажатии – надпись появляется.

5. Создать приложение, в котором подсчитываются события - нажатие левой клавиши мыши. А результат - число нажатий - выводится в секции WM\_PAINT только по нажатии правой клавиши мыши.

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

if (GetKeyState(VK\_RBUTTON)) {

char str[256] = "";

sprintf\_s(str, sizeof(str), "Number of pressing left button - %d", ReDrawNumber);

TextOut(hdc, 100, 100, str, strlen(str));

EndPaint(hWnd, &ps);

}

}

break;

case WM\_LBUTTONDOWN:

ReDrawNumber++;

InvalidateRect(hWnd, NULL, TRUE);

break;

Функция GetKeyState отвечает за возврат статуса кнопки клавиатуры или мышки, в котором старший бит равен 1 если нажата кнопка, иначе 0. Значение младшего бита 1 если кнопка включена(свойственно таким кнопкам как Capslock),иначе 0. Значению 0x80 как раз свойственно значение 1 только старшему биту.

6. Модифицировать приложение. Результат выводить в секции WM\_PAINT сразу при его обновлении. Для этого инициировать перерисовку окна функцией InvalidateRect ( hWnd, NULL, TRUE ).

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

char str[256];

sprintf\_s(str, "Number of pressing left button - %d", ReDrawNumber);

TextOut(hdc, 100, 100, str, strlen(str));

ValidateRect(hWnd, NULL);

EndPaint(hWnd, &ps);

}

break;

case WM\_LBUTTONDOWN:

ReDrawNumber++;

InvalidateRect(hWnd, NULL, true);

break;

Функция InvalidateRect запускает принудительную перерисовку окна, в нашем случае, при нажатии левой кнопки мыши окно будет перерисовано.

7. Создать приложение для вывода текста “Работает ТКП”, начиная с позиции X, Y. За начальное значение координат взять 0, 0. По сообщению WM\_LBUTTONDOWN увеличивать координаты точки вывода на 50 единиц и инициировать перерисовку. По сообщению WM\_RBUTTONDOWN уменьшать координаты точки вывода на 50 единиц и инициировать перерисовку.

case WM\_LBUTTONDOWN:

x += 50; y += 50;

InvalidateRect(hWnd, NULL, true);

break;

case WM\_RBUTTONDOWN:

x -= 50; y -= 50;

InvalidateRect(hWnd, NULL, true);

break;

case WM\_PAINT: {

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

sprintf\_s(str, "MoveME");

TextOut(hdc, x, y, str, strlen(str));

ValidateRect(hWnd, NULL);

EndPaint(hWnd, &ps);

}

break;

WM\_RBUTTONDOWN – отвечает за нажатие правой кнопки мыши.

8. Создать приложение для вывода координат указателя “мыши” в виде Х = …, У = ….. За начальное значение координат взять 0, 0. По сообщению WM\_LBUTTONDOWN считывать текущие координаты курсора мыши. По сообщению WM\_RBUTTONDOWN инициировать вывод координат. Обеспечивать перерисовку окна. Блокировать обновление вывода по другим причинам перерисовки.

case WM\_LBUTTONDOWN:

x = LOWORD(lParam);

y = HIWORD(lParam);

break;

case WM\_RBUTTONDOWN:

ReDraw = true;

InvalidateRect(hWnd, NULL, true);

break;

case WM\_PAINT: {

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

if (ReDraw) {

sprintf\_s(str, "x = %d,y = %d", x, y);

}

TextOut(hdc, 0, 0, str, strlen(str));

ValidateRect(hWnd, NULL);

EndPaint(hWnd, &ps);

ReDraw = false;

}

break;

Функции LOWORD и HIWORD позволяют получить координаты мыши.

**Вывод:** 1. Ознакомился со структурой и особенностями программирования оконных windows-приложений. 2. Изучил каркасы windows-приложений. 3. Изучил структуру типового каркаса оконных приложений (ТКП), использование ТКП для создания пользовательских приложений. 4. Ознакомился с особенностями организации вывода в оконных windows-приложениях. 5. Ознакомился с особенностями управления сообщениями.