

## Раздел «Язык Си» . OOP-GUI :

- Приложение wxWidgets
  - Видео
  - Примеры программ
  - 🍌 Задачи
    - 🍌 Задача 1
    - 🍌 Задача 2.
    - 🍌 Задача 3 Шашки на кольце
    - 🍌 Задача 4\*
- Требования к самостоятельно выбранному проекту

## Приложение wxWidgets

### Видео

- [Видео по теме](#)

### Примеры программ

Оконные приложения несколько отличаются от консольных программ.

Каждое оконное приложение работает с событиями, которые могут происходить со всеми его частями и устройствами.

Для обработки этих событий включается **цикл обработки событий**, который работает все время пока запущено приложение. Некоторые события приложение может отслеживать и обрабатывать.

Для работы с приложениями уже имеется достаточно большое количество классов, описывающее работу тех или иных устройств окна.

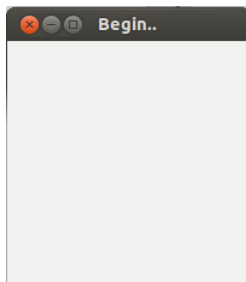
Рассмотрим пример самого простого приложения

```
#include <wx/wx.h>

// Уже имеется абстрактный класс wxApp для создания
// приложения
// наше приложение будет наследником этого wxApp
// виртуальную функцию OnInit необходимо определить в нашем классе
class Begin: public wxApp{
public:
    virtual bool OnInit();
};

// Макрос для запуска приложения (вместо main)
// В скобках указываем как называется наш класс-приложение
IMPLEMENT_APP(Begin)

// переопределяем виртуальную функцию OnInit
// именно она и запускает приложение
bool Begin::OnInit()
{
    // создаем динамический объект класса wxFrame (наше окно)
    //окно будет "пустое" размером 200x200 пикселей
    wxFrame *wind=new wxFrame(NULL,wxID_ANY,wxT("Begin.."),wxDefaultPosition,wxSize(200,200));
    // запуск. Окно будет видимое
    wind->Show(true);
    return true;
};
```



Расширим возможности нашего окна. Добавим в него меню.

```
#include <wx/wx.h>
// Класс нашего приложения
class Begin: public wxApp{
public:
```

Поиск

Поиск

## Раздел «Язык Си»

Главная

Зачем учить C?

Определения

Инструменты:

Поиск

Изменения

Index

Статистика

## Разделы

Информация

Алгоритмы

Язык Си

Язык Ruby

Язык Ассемблера

EJ Judge

Парадигмы

Образование

Сети

Objective C

Login&gt;&gt;

```

        virtual bool OnInit();
};

// Окошко было пустое. Поэтому расширим фрейм.
// Для этого создадим класс-наследник MyWin

class MyWin:public wxFrame{
// Добавим элементы:

    wxMenuBar *menubar; // полоска для меню
    wxMenu *file; // менюшка на полоске
    wxStatusBar *sb; // статус-бар
// Пункты меню для открытия файла и
// закрытия приложения
    wxMenuItem *load,*quit;
    wxString ss; // строка (пригодится)

public:
// конструктор с заголовком окна
    MyWin(const wxString& title);

// Функции, которые будут вызываться при выборе пунктов меню
    // Функция закрытия окна
    void OnQuit(wxCommandEvent& event);
// функция загрузки файла
    void OnLoad(wxCommandEvent& event);
};

// Идентификаторы нужны всем элементам, которые будут обрабатываться обработчиками событий
// идентификатор пункта меню
const int ID_MENU_LOAD =1002;

MyWin::MyWin(const wxString& title):wxFrame(NULL,wxID_ANY,title,wxDefaultPosition,wxSize(200,200)){
    // создали полосу для менюшки
    menubar = new wxMenuBar;
// создали менюшку
    file = new wxMenu;
    quit = new wxMenuItem(file, wxID_EXIT, wxT("&Quit"));
    load = new wxMenuItem(file, ID_MENU_LOAD, wxT("&Load"));
    file->Append(load);
    file->Append(quit);
// закинули менюшку на полосу
    menubar->Append(file, wxT("&File"));
// установили полосу в окно
    SetMenuBar(menubar);

// Connect служит для соединения элемента приложения с обработчиком событий
// подключили менюшку
// wxID_EXIT - стандартный идентификатор для выключения приложения,
// wxEVT_COMMAND_MENU_SELECTED - идентификатор действия (выбор пункта меню)
// wxCommandEventHandler - обработчик событий, связанных с командами: меню, кнопки...
// MyWin::OnQuit - функция, которая будет вызываться при этом событии
    Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(MyWin::OnQuit));
// ID_MENU_LOAD - не стандартный идентификатор (сами определяли)
// MyWin::OnLoad - функция, которая будет вызываться при этом событии
    Connect(ID_MENU_LOAD, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(MyWin::OnLoad));

    sb=CreateStatusBar();
    sb->SetStatusText(wxString(wxT("что-то напишем и здесь")));
};

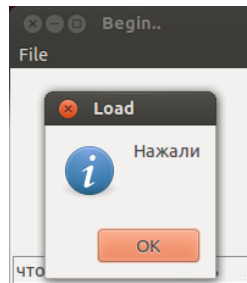
// Функция выключения окна
// Параметр - объект класса wxCommandEvent
// Можно использовать его методы
// Но здесь не нужно
void MyWin::OnQuit(wxCommandEvent& event){
    Close(true);
};

// Функция для загрузки
void MyWin::OnLoad(wxCommandEvent& event){
// Ничего пока не происходит
// просто будет сообщение
    wxMessageBox(wxT("Нажали"),wxT("Load"));
};

IMPLEMENT_APP(Begin)

bool Begin::OnInit()
{
    MyWin *wind=new MyWin(wxT("Begin.."));
    wind->Show(true);
    return true;
};

```



А теперь попробуем написать что-то разумное

Это приложение будет открывать файл с координатами прямоугольника, показывать их в текстовом окне и рисовать этот прямоугольник

```
#include <wx/wx.h>
#include <wx/file.h>
#include <wx/wfstream.h>
#include <wx/txtstrm.h>
#include <wx/sstream.h>
#include <wx/string.h>

class Begin: public wxApp{
public:
    virtual bool OnInit();
};
// Нужен еще класс Draw
class Draw;
class MyWin;

class MyWin:public wxFrame{
    wxMenuBar *menubar; // полоска для меню
    wxMenu *file,*im; // менюшка на полоске
    wxMenuItem *load,*quit; // открывалка файла
    wxTextCtrl *tc; // текстовое окошко
    // Указатель на объект Draw
    Draw *dp;
    // панель
    // если элементы помещать сразу на фрейм, то первый же объект
    // займет весь фрейм
    // для различных элементов есть wxPanel
    wxPanel *m_pan;
    // Это в "подвале" окошка
    wxStatusBar *sb; // статус бар
    // строка (для всех типов кодировки)
    wxString ss;
public:
    // конструктор
    MyWin(const wxString& title);
    // две координаты для рисования
    wxPoint a;
    wxPoint b;
    // Функция закрытия окна
    void OnQuit(wxCommandEvent& event);
    // Загрузка файла с данными
    void OnLoad(wxCommandEvent& event);
};

// класс для рисования
// наследник wxPanel
class Draw: public wxPanel{
    // указатель на верхнее окно
    // это нужно для доступа к элементам (к а и b)
    MyWin *mn;
public:
    // в конструкторе указывается адрес объекта, который ее содержит
    Draw(wxPanel *parent, MyWin *main);
    // рисовалка
    void OnPaint(wxPaintEvent & event);
};

//идентификаторы
// загрузка
const int ID_MENU_LOAD =1002;
// редактирование
const int ID_MENU_EDIT =1003;

// Конструктор фрейма
MyWin::MyWin(const wxString& title):wxFrame(NULL,wxID_ANY,title,wxDefaultPosition,wxSize(400,400)){
    // создали полоску для менюшки
    menubar = new wxMenuBar;
    // создали менюшку
    file = new wxMenu;
```

```

// закинули менюшку на полосу
// пока не работает (для "красоты")

file->Append(wxID_ANY, wxT("&New"));

file->AppendSeparator();
// Для всех пунктов меню указываем идентификатор чтобы
// связать обработчик события с конкретным элементом
// добавили к менюшке раздел quit
quit = new wxMenuItem(file, wxID_EXIT, wxT("&Quit"));
// добавили раздел load
load = new wxMenuItem(file, ID_MENU_LOAD, wxT("&Load"));
file->Append(load);
file->Append(quit);
// Это еще один пункт меню
// пока тоже не работает
im = new wxMenu;
im->Append(wxID_ANY, wxT("Edit figure"));
im->Append(wxID_ANY, wxT("Rotate 90"));
// добавили оба пункта меню на полосу меню
menubar->Append(file, wxT("&File"));
menubar->Append(im, wxT("&Edit"));

// установили полосу в окно
SetMenuBar(menubar);
// подключили менюшку exit
Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(MyWin::OnQuit));
// подключили load (идентификаторы такие же как и при создании каждого элемента)
Connect(ID_MENU_LOAD, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(MyWin::OnLoad));

// создание панельки для текста, кнопок и рисовалки
m_pan = new wxPanel(this, wxID_ANY);
// это тоже панель, но наша.
// помещаем ее на панель m_pan и задаем указатель на главный фрейм
dp = new Draw(m_pan, this);
// это окошко для текста. Также помещаем на m_pan
tc = new wxTextCtrl(m_pan, -1, wxT(""), wxPoint(200, 10), wxSize(200, 50));
// статус-бар будет внизу окна
sb = CreateStatusBar();
sb->SetStatusText(wxString(wxT("что-то напишем и здесь")));
};

void MyWin::OnLoad(wxCommandEvent& event){
// специальный класс для листания файлов
wxFileDialog * openFileDialog = new wxFileDialog(this);
// если все открывается, выберем имя файла
// Только имя!!!
if (openFileDialog->ShowModal() == wxID_OK){
// Что выбрали, то и будет именем файла
// Запоминаем в строку
wxString fileName = openFileDialog->GetPath();
// Загружаем содержимое в окно текста
tc->LoadFile(fileName);
// Теперь нужно получить данные из файла
// Создаем объект - файловый поток
wxFileInputStream input(fileName);
// Чтобы он работал как текстовый файл, превращаем его в поток-текст
wxTextInputStream intext(input);
int x1, y1, x2, y2;
// "обычным" образом считываем данные
intext>>x1>>y1>>x2>>y2;
// Это были координаты
a.x = x1;
a.y = y1;
b.x = x2;
b.y = y2;
// А еще можно сделать поток из строки
// но к работе окна это отношения не имеет
wxString a;
// В строку записали текст
a<<wxT("123 17 89 12.5");
// превратили в поток
wxStringInputStream st(a);
// превратили в поток-текст
wxTextInputStream in(st);
int a1, a2, a3;
float w;
// получили данные
in>>a1>>a2>>a3>>w;
// Это нужно для перерисовки окна, когда будем рисовать фигуру
dp->Refresh();
}
};

```

```

void MyWin::OnQuit(wxCommandEvent& event){
    Close(true);
};

// Конструктор нашего Draw
Draw::Draw(wxPanel *parent, MyWin *fr):wxPanel(parent, -1,wxPoint(50,50),wxSize(100,100),wxBORDER_SUNKEN){
    // подключили панель к событиям рисования
    Connect(wxEVT_PAINT,wxPaintEventHandler(Draw::OnPaint));
    mn = fr;
};

// Свободная функция рисования прямоугольника
// Для рисования есть абстрактный класс wxDC
// У него много разных наследников
// Самый простой wxPaintDC
// Но мы сделаем функцию для всех наследников wxDC
void DrRec(wxPoint a, wxPoint b, wxDC * dc){
    dc->DrawRectangle(a, wxSize (abs(a.x-b.x),abs(a.y-b.y)));
};

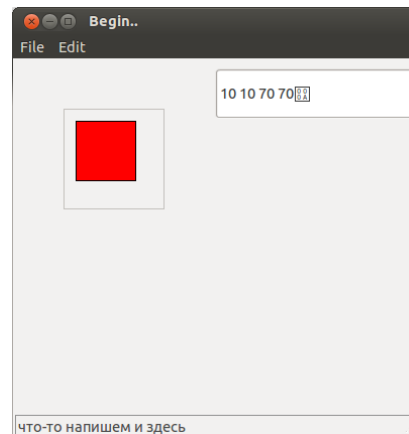
// Функция OnPaint() сработает при любом рисовании окна:
// первое рисование, сложил-разложили
// или вызов функции Refresh()

void Draw::OnPaint(wxPaintEvent& event){
    // положили планшет wxPaintDC на нашу панель
    wxPaintDC dc(this);
    // Установили цвет заливки
    dc.SetBrush(wxBrush(wxColour(255,0,0)));
    // передали свободной функции координаты и указатель на планшет
    DrRec(mn->a,mn->b,&dc);
};

// запуск окна
IMPLEMENT_APP(Begin)

bool Begin::OnInit()
{
    MyWin *wind=new MyWin(wxT("Begin.."));
    wind->Show(true);
    return true;
};

```



## 🔧 Задачи

### 🔧 Задача 1

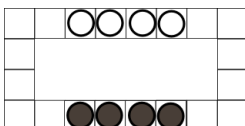
В файле указано несколько прямоугольников и цвет для каждого. Загрузить файл и нарисовать все

### 🔧 Задача 2.

Объявить абстрактный класс **Figura** с виртуальной функцией **void draw()**. В конструктор передается указатель на планшет. Написать наследники **Circle**, **Triangle** и **Rectangle** Их параметры задаются в файле. Загрузить файл, нарисовать все фигуры в том порядке как они указаны в файле

### 🔧 Задача 3 Шашки на кольце

Дано кольцо, размером  $N=4 \times n$ ,  $1 \leq n \leq 100$  клеточек и  $k=N/4$  белых и черных шашек. Кольцо делится пополам и шашки выставляются друг напротив друга.



Игроки ходят по-очереди. Белые первые. Каждый игрок может двигать шашку своего цвета на одну свободную клетку вправо или влево. Если во время хода на соседней с шашкой клетке стоит шашка

другого цвета, а за ней пустая клетка, шашку нужно "съесть". Если после "соединения" рядом опять оказалась шашка другого цвета, то таким образом "соедаются" все такие шашки.

Цель игры: "съесть" все шашки другого цвета, оставив хотя бы одну свою.

Реализовать игру как GUI-приложение.. Поле – контейнер. Шашки – объекты с заданными свойствами хода.

#### **Задача 4\***

Про моделировать процесс столкновения двух абсолютно упругих шариков с единичной массой и заданными диаметром и скоростью в коробке заданного размера. Пример посмотреть в папке sharik.

Документация [wxWidgwets.pdf](#), [docs.wxwidgets.org](#), [zetcode.com](#)

Примеры задач для реализации как GUI-приложение:


[intrigi.pdf](#)

[robot\\_gonki.pdf](#)

### **Требования к самостоятельно выбранному проекту**

1. Проект должен быть реализован как GUI-приложение на языке C++ с использованием wxWidgets (чтобы задачи не были позаимствованы)
2. Проект должен представлять: а) пошаговую игру; б) эмулятор несложного устройства (калькулятор, робот и т.д.); в) модель физического или математического явления
3. Проект должен быть распределен не менее чем на 2 человека.
4. Постановка задачи для проекта должна быть написана на русском языке и представлена до 25 марта
5. Проект должен быть разделен на модули, которые пишутся каждым участником отдельно. Каждый модуль тестируется и сдается отдельно. Затем рассматривается собранный целый проект. При описании необходимо четко описать способы взаимодействия модулей – форматы описания и передачи данных.

-- [TatyanaOvsyannikova2011](#) – 15 Apr 2016

Attachment 	Action	Size	Date	Who	Comment
 win.png	<a href="#">manage</a>	4.5 K	15 Apr 2016 - 18:28	<a href="#">TatyanaOvsyannikova2011</a>	
 win2.png	<a href="#">manage</a>	12.1 K	15 Apr 2016 - 18:54	<a href="#">TatyanaOvsyannikova2011</a>	
 wxWidgets.pdf	<a href="#">manage</a>	6963.7 K	15 Apr 2016 - 19:57	<a href="#">TatyanaOvsyannikova2011</a>	
 win3.png	<a href="#">manage</a>	10.9 K	15 Apr 2016 - 20:01	<a href="#">TatyanaOvsyannikova2011</a>	
 robot_gonki.pdf	<a href="#">manage</a>	116.8 K	05 Apr 2017 - 17:13	<a href="#">TatyanaOvsyannikova2011</a>	
 intrigi.pdf	<a href="#">manage</a>	73.4 K	05 Apr 2017 - 17:13	<a href="#">TatyanaOvsyannikova2011</a>	
 cshash.png	<a href="#">manage</a>	12.0 K	19 Apr 2017 - 15:58	<a href="#">TatyanaOvsyannikova2011</a>	

(с) Материалы раздела "Язык Си" публикуются под лицензией [GNU Free Documentation License](#).