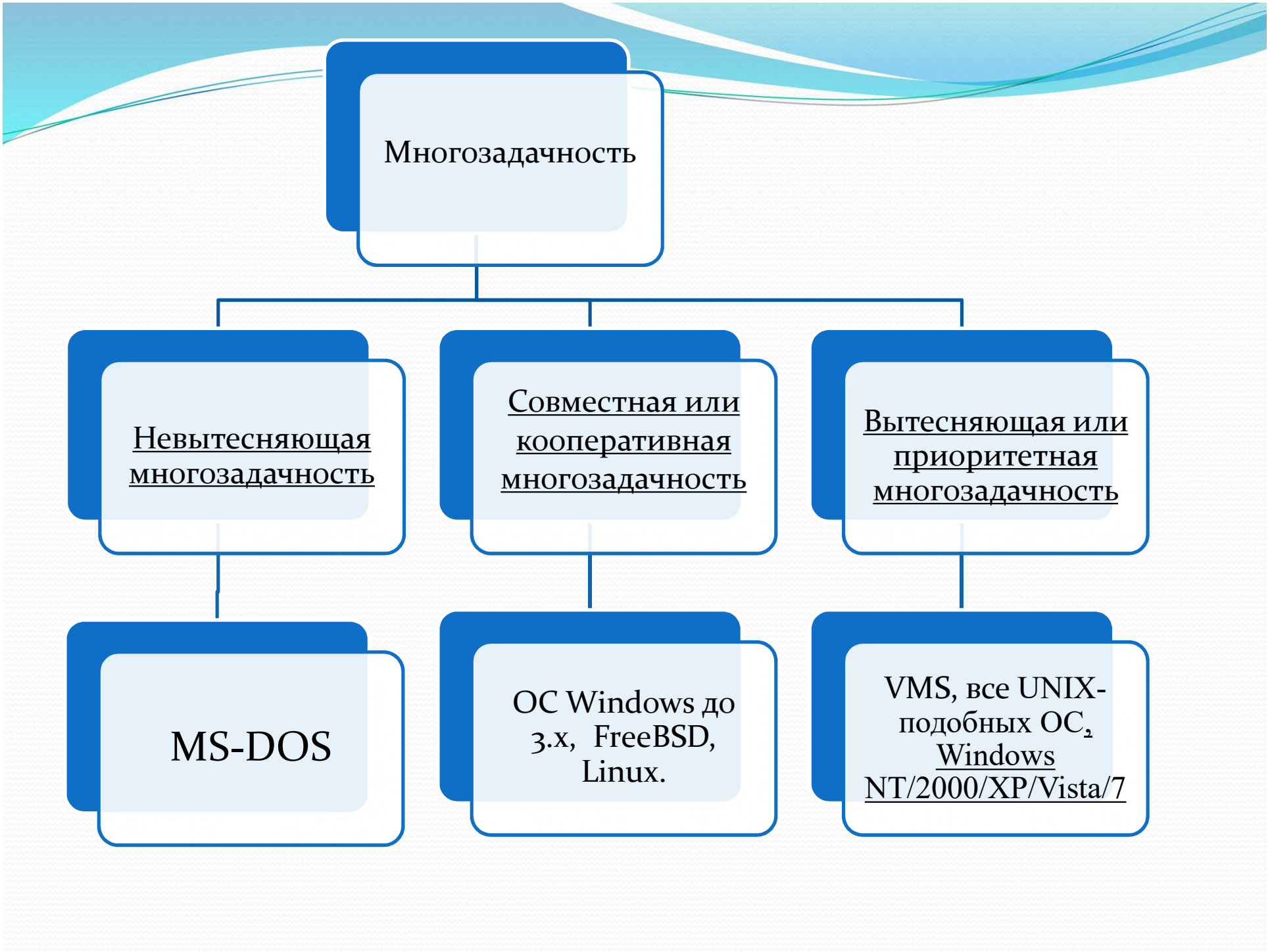
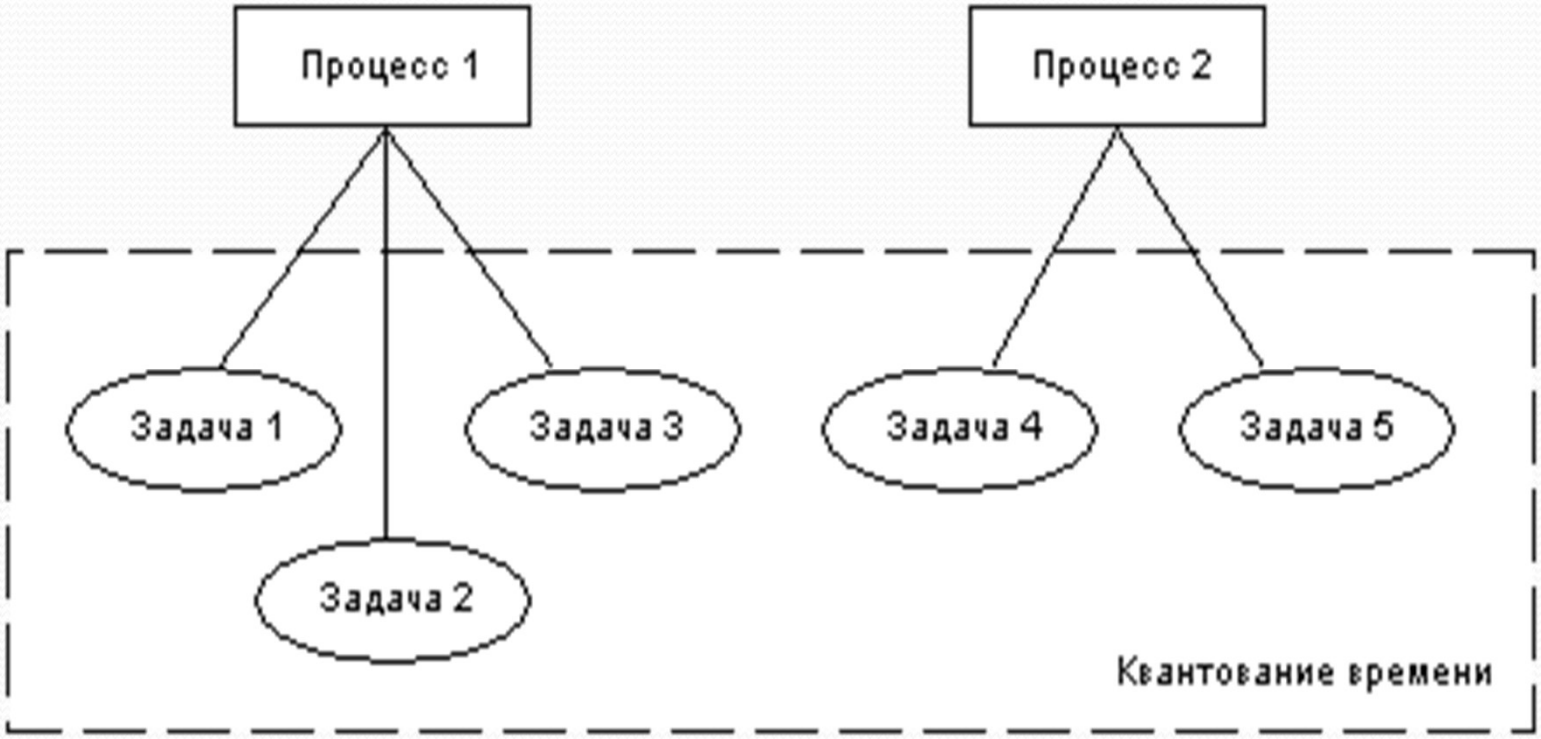
Основы программирования для Windows

Особенности ОС Windows

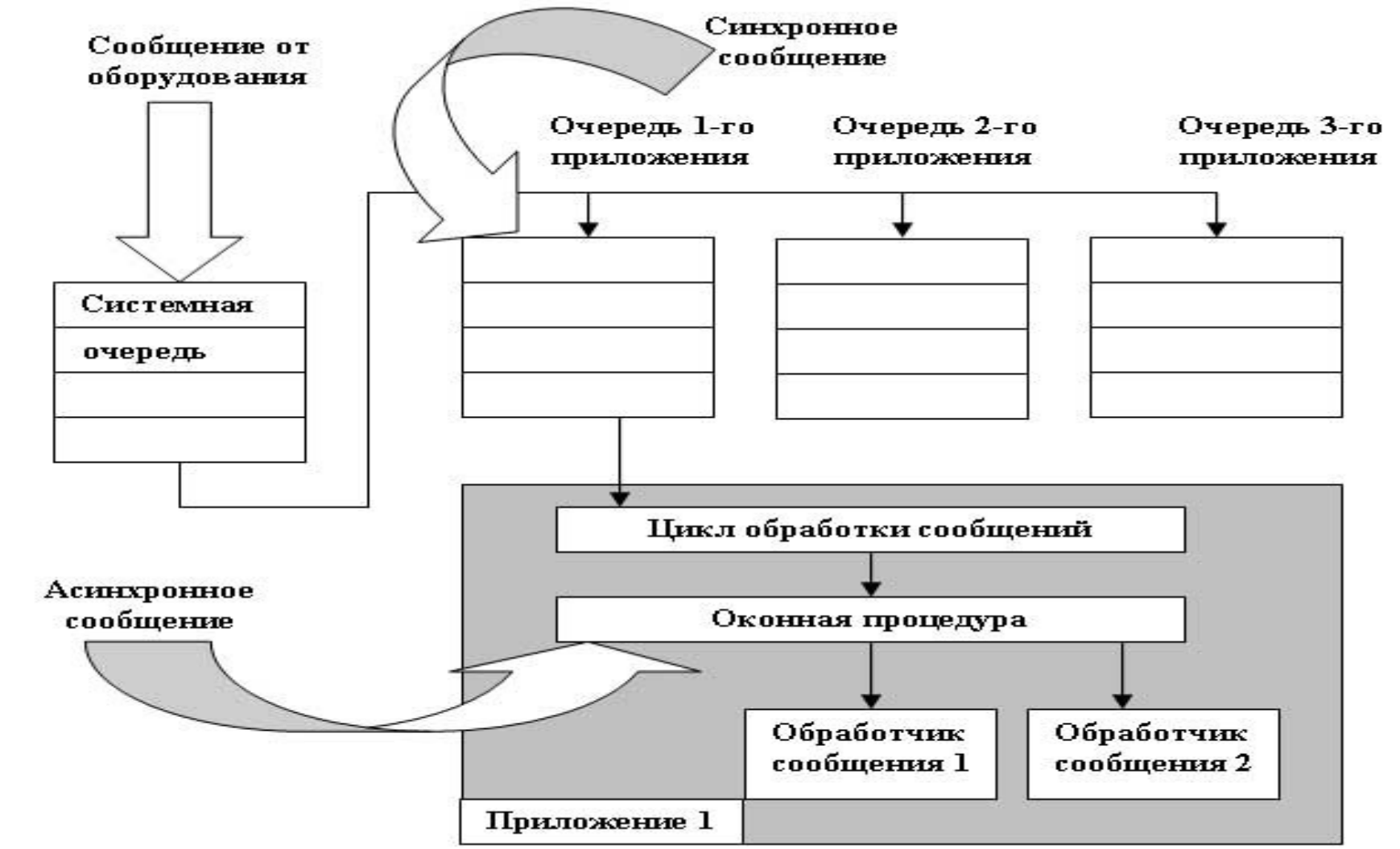
- Файловая система использует таблицу размещения файлов защищенного режима (Protected-mode FAT или VFAT).
- Вытесняющая многозада́чность.
- Несегментированная модель памяти FLAT-модель.
- Библиотеки динамической загрузки.
- Очереди сообщений
- Регистрационная база данных


Многозадачность
Невытесняющая многозадачность
Совместная или кооперативная многозадачность
Вытесняющая или приоритетная многозадачность
MS-DOS
ОС Windows до 3.x, FreeBSD, Linux.
VMS, все UNIX-подобных ОС, Windows NT/2000/XP/Vista/7

Процесс и поток



Очереди сообщений

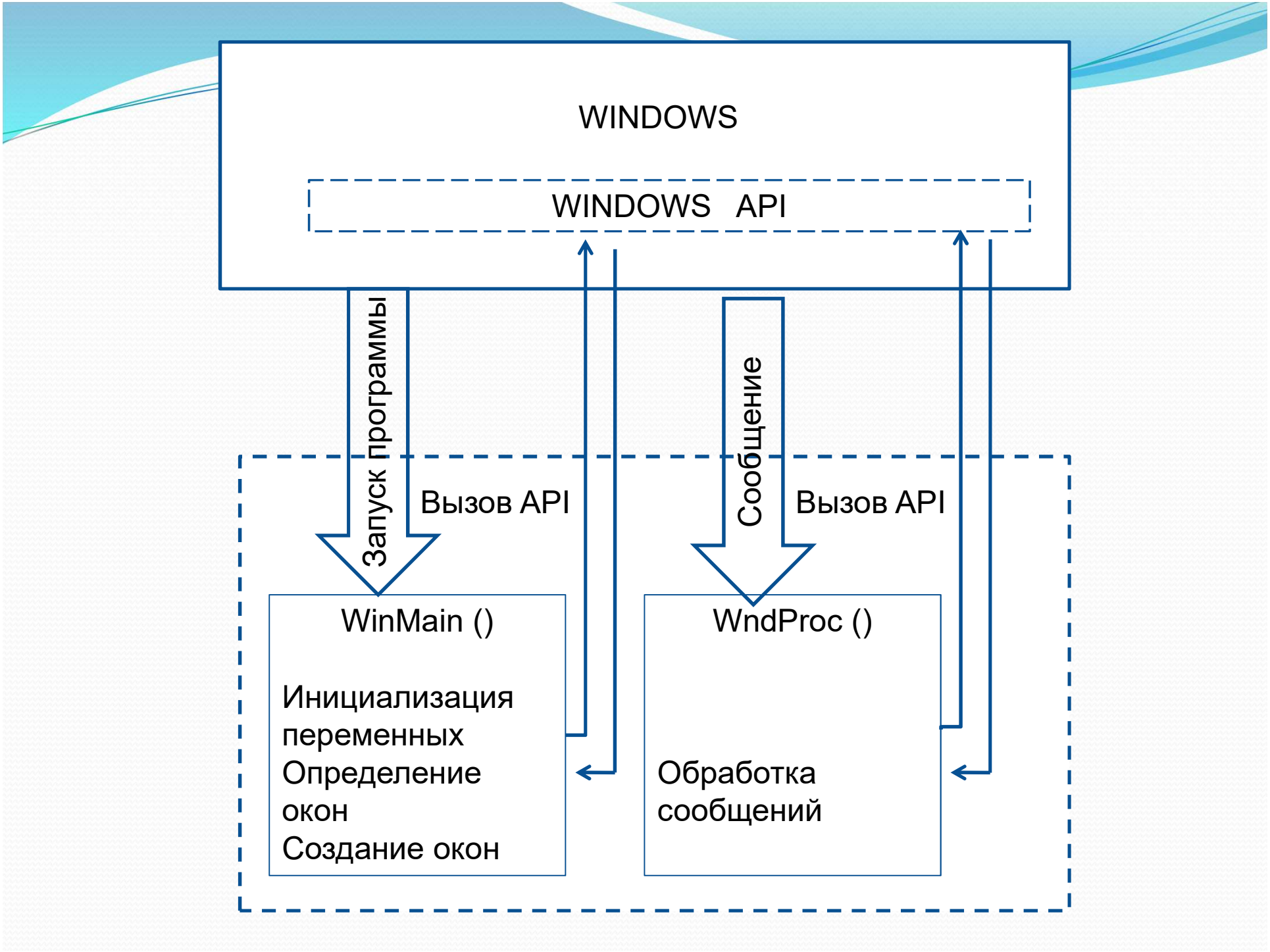


Структура сообщения

```
typedef struct MSG
{
 HWND        hwnd;
 UINT        message;
 WPARAM      wParam;
 LPARAM      lParam;
 DWORD       time;
 POINT       pt;
}
```

Способы создания интерактивных Win приложений в VS C++

- Использование интерфейса WIN API
- Использование классов MFC.
- Использование Windows Forms.


WINDOWS
WINDOWS API
Запуск программы
Вызов API
Сообщение
Вызов API
WinMain ()
Инициализация переменных
Определение окон
Создание окон
WndProc ()
Обработка сообщений

Функция WinMain

```
int  APIENTRY  WinMain  (HINSTANCE  hIinstance,
                          HINSTANCE  hPrevInstance,
                                LPSTR      lpszCmdLine,
                                int        nCmdShow)
                                //SW_MINIMIZE, SW_SHOW
{
```

- Начальная инициализация приложения (подготовка данных класса окна и его регистрация);
- Создание главного окна приложения
- Запуск цикла обработки сообщений, извлекаемых из очереди

```
}
```

Регистрация класса окна

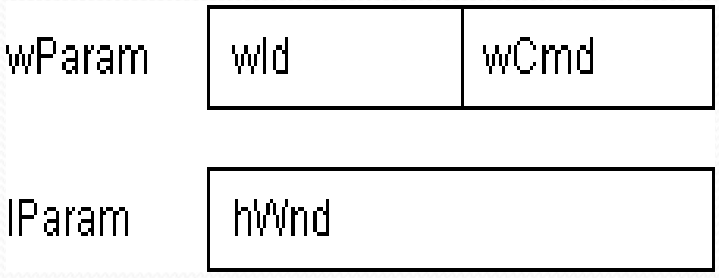
```
 WNDCLASSEX wc;
wc.cbSize = sizeof(WNDCLASSEX);
wc.hIconSm = 0 ;
 wc.style = CS_HREDRAW|CS_VREDRAW;
wc.lpfnWndProc = (WNDPROC)WndProc;
wc.cbClsExtra  = 0;
wc.cbWndExtra  = 0;
 wc.hInstance =hInst;
wc.hIcon = LoadImage(hInst,
                     MAKEINTRESOURCE(IDI_APPLICATION));
 wc.hCursor = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH) (COLOR_WINDOW + 1);
wc.lpszMenuName = NULL;
wc.lpszClassName = szAppName;
if(!RegisterClassex(&wc))
    return FALSE;
```

Создание окна

```
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{ HWND hWnd;
hWnd = CreateWindow(szWindowClass,
                    szTitle,
                    WS_OVERLAPPEDWINDOW,//стиль окна
                    CW_USEDEFAULT, 0, // координата левого верхнего угла
                    CW_USEDEFAULT, 0, // ширина, высота окна
                    NULL, // дескриптор родительского окна
                    NULL, // дескриптор меню окна
                    hInstance,    // дескриптор экземпляра приложения
                    NUL)  ;   // указатель на дополнительны данные окна;
  if (!hWnd)
  {   return FALSE;   }
  ShowWindow(hWnd, nCmdShow);
  UpdateWindow(hWnd);
  return TRUE;
}
```

Функция окна

```
LRESULT WINAPI      WndProc(HWND hWnd,
                            UINT msg,
                            WPARAM wParam,
                            LPARAM lParam);
LRESULT CALLBACK    WndProc(HWND hWnd,
                            UINT msg,
                            WPARAM wParam,
                            LPARAM lParam);
```



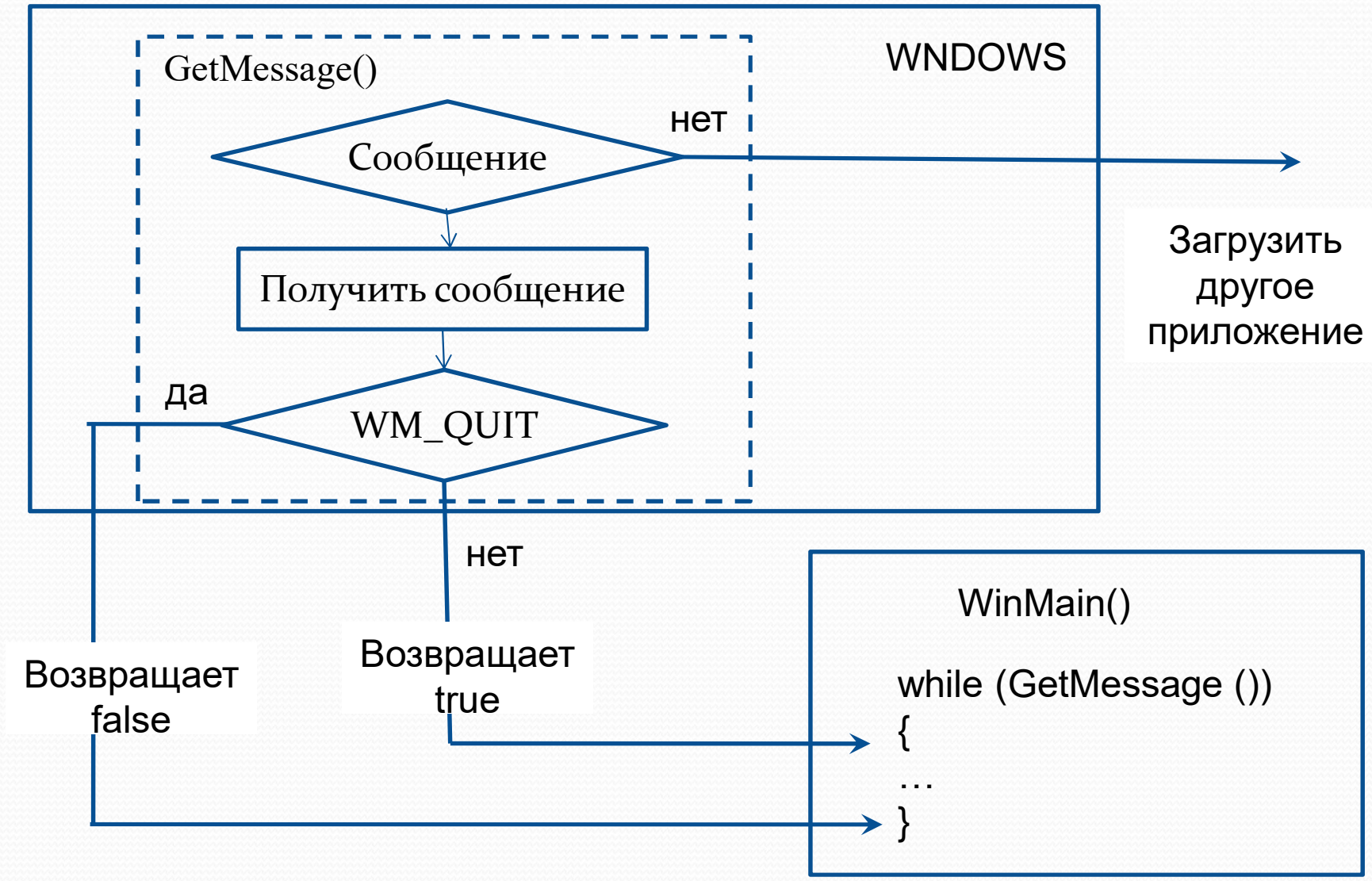
Функция окна

```
LRESULT          WINAPI    WndProc(HWND hWnd, UINT msg,
                           WPARAM wParam, LPARAM lParam)
{
switch(msg)
  {  case WM_LBUTTONDOWN:
     {       MessageBox(NULL,"Нажата левая клавиша мыши«,"Сообщение ",
                                   MB_OK|MB_ICONINFORMATION);
             return 0;
     }
              }
     case WM_DESTROY:
     {
              PostQuitMessage(0);
              return 0;
     }

             default:  DefWindowProc(hwnd, msg, wParam, lParam);
}
```

WM_QUERYENDSESSION

Концептуальная структура функции GetMessage()



Цикл обработки сообщения

```
MSG msg
while (GetMessage (&msg,NULL,0,0))
{   TranslateMessage(&msg);
    DispatchMessage(&msg)
    }
```

Обрабатываемые сообщения:

При создании окна	WM_CREATE, WM_GETMINMAXINFO WM_NCCREATE
При перерисовке	WM_PAINT
При выходе	WM_QUIT
При завершении работы Windows	WM_QUERYENDSESSION

Определение запущенной копии приложения

```
HWND FindWindow
   (  LPCTSTR lpClassName,
   LPCTSTR lpWindowName);

BOOL IsIconic  (HWND hwnd);

BOOL SetForegroundWindow  (HWND hwnd);
```

Пример приложения

```
    LRESULT WINAPI    WndProc(HWND hWnd, UINT msg,
                                        WPARAM wParam, LPARAM lParam);
    szAppName[] = "Window";
    char szAppTitle[] = "Window Application";

int  APIENTRY  WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                        LPSTR lpCmdLine, int nCmdShow)
{   WNDCLASSEX wc;
    HWND hWnd;
    MSG msg;
    hInst = hInstance;
    hWnd = FindWindow(szAppName, NULL);
    if   (hWnd)
    { if   (IsIconic(hWnd))
    ShowWindow(hWnd, SW_RESTORE);
    SetForegroundWindow(hWnd);
    return FALSE;
     }
```

Создание главного окна приложения и цикл обработки сообщений

```
hWnd = CreateWindow(szAppName,
                    szAppTitle,
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT,  CW_USEDEFAULT,
                    NULL, NULL,
                    hInst,
                  NULL);
 if(!hWnd) return(FALSE);
  ShowWindow(hWnd, nCmdShow);
 UpdateWindow(hWnd);
  while(GetMessage (&msg, NULL, 0, 0))
 {    TranslateMessage(&msg);
     DispatchMessage(&msg);
 }
return msg.wParam;
}
```