Белорусский государственный университет
Факультет прикладной математики и информатики
Кафедра технологии программирования
доцент Побегайло А.П.

1. Операционные системы и интерфейсы пользователя.

1.1. Интерфейс программирования приложений Win32 API.

Интерфейс программирования приложений Win32 API представляет собой набор функций и классов, которые используются для программирования приложений на базе операционных систем фирмы Microsoft. Следует отметить, что в работе многих функций Win32 API существуют различия, которые зависят от типа операционной систем. Кроме того, некоторые функции работают только в операционной системе Windows 2000 и не поддерживаются операционной системой Windows 98. Все эти случаи будут отмечаться отдельно. Но все же в работе функций Win32 API на разных операционных системах гораздо больше общего, чем различий. Поэтому, чаще всего мы будем говорить, что функции Win32 API предназначенных для разработки приложений для операционных систем Windows, не делая различия между операционными системами Windows 98 и Windows 2000. Это соглашение значительно облегчит изложение, не загромождая его ненужными подробностями, которые отвлекают от главного смысла излагаемого материала.

Функционально, Win32 API подразделяется на следующие категории:

– Base Services (базовые сервисы);
– Common Control Library (библиотека общих элементов управления);
– Graphics Device Interface (интерфейс графических устройств);
– Network Services (сетевые сервисы);
– User Interface (интерфейс пользователя);
– Windows NT Access Control (управление доступом для Windows NT);
– Windows Shell (оболочка Windows);
– Windows System Information (информация о системе Windows).

Кратко опишем функции, которые выполняются в рамках этих категорий. Функции базовых сервисов обеспечивают приложениям доступ к ресурсам компьютера. Категория Common Control Library содержит классы окон, которые часто используются в приложениях. Интерфейс графических устройств обеспечивает функции для вывода графики на дисплей, принтер и другие графические устройства. Сетевые сервисы используются для соединения компьютеров в компьютерные сети. Интерфейс пользователя обеспечивает функции для взаимодействия пользователя с приложением, используя окна для ввода-вывода информации. Категория Windows NT Access Control содержит функции, которые используются для защиты информации посредством управления доступом к ней. Категории Windows Shell и Windows System Information содержат соответственно функции для работы с оболочкой и конфигурацией операционной системы Windows.

В курсе операционные системы главным образом изучается назначение и использование функций из категорий Base Services, а сам курс часто называется системное программирование. Функций из категорий Common Control Library, Graphics Device Interface и User Interface используются для разработки интерфейса приложений, а курс, который изучает назначение и использование этих функций, часто называется программирование пользовательских интерфейсов в Windows. Изучив эти два курса и добавив сюда свои знания по программированию на языке C++, вы получите довольно содержательное представление о разработке приложений в среде операционной системы Windows.

В связи с тем, что программирование графических пользовательских интерфейсов в Windows само по себе является довольно трудоемким занятием, мы будем изучать функции ядра Windows, работая только с консольными приложениями. Это упростит изложение предмета и избавит нас от большого количества кода, не относящегося к существу рассматриваемых вопросов.

1.2. Типы данных в Win32 API

Прежде всего заметим, что интерфейс программирования приложений Win32 API ориентирован на язык программирования C или в более широком смысле на процедурные языки программирования. Поэтому в этом интерфейсе не используются такие возможности языка программирования C++, как классы, ссылки и механизм обработки исключений.

Чтобы сделать интерфейс Win32 API более независимым от конкретного языка программирования или, может быть, более соответствующим аппаратному обеспечению компьютера, разработчики этого интерфейса определили новые простые типы данных. Эти типы данных используются в прототипах функций интерфейса Win32 API.

Новые простые типы данных определены как синонимы простых типов данных языка программирования C. Чтобы отличать эти типы от других типов, их имена определены прописными буквами. Общее количество простых типов данных, определенных в интерфейсе Win32 API, довольно велико. Поэтому ниже приведены определения только тех простых типов данных из этого интерфейса, которые очевидным образом переименовывают простые типы данных языка программирования C.

```
typedef unsigned char  UCHAR;
typedef UCHAR          *PUCHAR;
typedef unsigned char  BYTE;
typedef BYTE           *PBYTE;
typedef BYTE           *LPBYTE;

typedef unsigned short  USHORT;
typedef USHORT          *PUSHORT;
typedef unsigned short  WORD;
typedef WORD            *PWORD;
typedef WORD            *LPWORD;

typedef int   INT;
typedef int   *PINT;
typedef int   *LPINT;
typedef int   BOOL;
typedef BOOL  *PBOOL;
typedef BOOL  *LPBOOL;

typedef unsigned int  UINT;
typedef unsigned int  *PUINT;

typedef long   *LPLONG;

typedef unsigned long  ULONG;
typedef ULONG           *PULONG;
typedef unsigned long   DWORD;
typedef DWORD           *PDWORD;
typedef DWORD           *LPDWORD;

typedef float  FLOAT;
typedef FLOAT  *PFLOAT;

typedef void   *LPVOID;
typedef CONST void  *LPCVOID;
```

Остальные простые типы данных, определенные в интерфейсе Win32 API, имеют, как правило, специфическое назначение и поэтому они будут описаны при их использовании.

Кроме того, в интерфейсе Win32 API определены символические константы `FALSE` и `TRUE` для обозначения соответственно ложного и истинного логических значений. Определения этих констант приведены ниже.

```
#ifndef FALSE
#define FALSE  0
#endif

#ifndef TRUE
#define TRUE  1
#endif
```

В интерфейсе Win32 API также определено множество сложных типов данных, таких как структуры и перечисления. Как правило, эти типы данных имеют специфическое назначение и поэтому будут описаны при их непосредственном использовании.

1.3. Объекты и их дескрипторы в Windows.

Объектом в Windows называется структура данных, которая представляет системный ресурс. Таким ресурсом может быть, например, файл, поток, графический рисунок. Операционные системы Windows предоставляют приложению объекты трех категорий:
– User (объекты интерфейса пользователя);
– Graphics Device Interface (объекты интерфейса графических устройств);
– Kernel (объекты ядра).

Категория User включает объекты, которые используются приложением для интерфейса с пользователем. К таким объектам относятся, например, окна и курсоры. Категория Graphics Device Interface включает объекты, которые используются для вывода информации на графические устройства. К таким объектам относятся, например, кисти и перья. Категория Kernel включает объекты ядра операционной системы Windows. К таким объектам относятся, например, потоки и каналы. В курсе операционные системы мы подробно рассмотрим только объекты категории Kernel. Объекты двух оставшихся категорий рассматриваются в курсе программирование интерфейсов в Windows.

Под *доступом к объектам* понимается возможность приложения выполнять над объектом некоторые функции. Приложение не имеет прямого доступа к объектам. Для того чтобы получить этот доступ, необходимо использовать его дескриптор (handle). В Win32 API дескриптор имеет тип HANDLE. *Дескриптор объекта* представляет собой запись в таблице, которая поддерживается системой и содержит адрес объекта и средства для идентификации типа объекта. Дескрипторы объектов создаются функциями Win32 API, которые, за редким исключением, имеют вид *CreateX*, где буква *X* заменяет название объекта. Как правило, такие функции возвращают дескриптор созданного объекта. Если это значение не равно NULL, то объект создан успешно.

После завершения работы с объектом его дескриптор нужно закрыть, используя функцию *CloseHandle*, которая имеет следующий прототип:

```
BOOL CloseHandle(
      HANDLE      hObject           // дескриптор объекта
);
```

При успешном завершении функция *CloseHandle* возвращает значение TRUE, в противном случае – FALSE. Функция *CloseHandle* удаляет дескриптор объекта, но сам объект удаляется не всегда. Дело в том, что в Windows на один и тот же объект могут ссылаться несколько дескрипторов, которые создаются другими функциями для доступа к уже созданному ранее объекту. Функция *CloseHandle* уничтожает объект только в том случае, если на него больше не ссылается ни один дескриптор.