Лабораторная работа№6

Задание 1 Задать вид курсора в пределах окна

**Windows** предоставляет набор стандартных курсоров, которые являются доступными для любой прикладной программы, чтобы использовать в любое время. Заголовочные файлы **Windows** содержат идентификаторы для стандартных курсоров - идентификаторы начинаются с префикса "**IDC\_**". Каждый стандартный курсор имеет соответствующее заданное по умолчанию изображение, связанное с ним. Пользователь или прикладная программа могут заменять заданное по умолчанию изображение, связанное с любым стандартным курсором в любое время. Прикладная программа заменяет заданное по умолчанию изображение, используя функцию **SetSystemCursor**. Прикладная программа может использовать функцию **GetIconInfo**, чтобы изъять информацию о текущем изображении курсора и может рисовать курсор на экране, используя функцию **DrawIconEx**. Чтобы нарисовать заданное по умолчанию изображение стандартного курсора при обращении к функции **DrawIconEx** определите флажок **DI\_COMPAT**. Если вы не определите флажок **DI\_COMPAT**, **DrawIconEx** рисует стандартный курсор, использующий изображение, которое определил пользователь.

Пользовательские курсоры разрабатываются для использования в специфической прикладной программе и могут быть любого внешнего вида, определяемого разработчиком.

Курсоры могут быть или одноцветными или цветными, и или статическими или "живыми". Тип курсора, используемого в отдельной компьютерной системе, зависит от системного дисплея. Старые дисплеи типа **VGA** не поддерживают цветные или "живые" указатели, однако новые дисплеи (чьи драйверы изображения используют механизм **DIB** (ФОРМАТ РАСТРОВОГО ИЗОБРАЖЕНИЯ)) поддерживают их.

Курсоры и иконки похожи и могут использоваться для взаимозаменяемости во многих ситуациях. Единственное различие между ними то, что изображение, заданное как курсор должно быть в формате, который дисплей может поддерживать. Например, курсор должен быть одноцветным для дисплея **VGA**.

**ОСТРИЕ (HOT SPOT)**

В курсоре, пиксель называемый острием (**hot spot**), отмечает точное экранное расположение, на которое воздействует событие мыши, типа щелчка ее кнопки. Как правило, острие - фокус курсора. Система устанавливает и распознает эту отметку как позицию курсора. Например, типичные острия - пиксель на кончике курсора имеющего форму стрелки и пиксель в середине курсора имеющего форму креста.

Когда происходит событие ввода информации от мыши, драйвер мыши **Windows** транслирует событие в соответствующее сообщение мыши, которое включает координаты острия. **Windows** посылает сообщение мыши окну, которое содержит острие или окну, которое захватило ввод информации от мыши. Для получения дополнительной информации, см. статью Ввод информации от мыши.

**МЫШЬ И КУРСОР**

**Windows** отражает движение мыши, перемещая соответственно курсор на экране. Когда курсор перемешается над различными частями окон или в различных окнах, **Windows** (или прикладная программа) изменяет вид курсора. Например, когда курсор пересекает границу окна, **Windows** изменяет курсор в двунаправленную стрелку.

Если система не имеет мыши, **Windows** показывает на экране и перемещает курсор только тогда, когда пользователь выбирает некоторые системные команды, типа тех, которые используются для установки размера или перемещения окна. Чтобы обеспечить пользователя способом отображения и перемещения курсора, когда мышь не доступна, прикладная программа, может использовать функции курсора, чтобы моделировать движение мыши. Получив эту возможность моделирования, пользователь может использовать клавиши со стрелками на клавиатуре, чтобы перемещать курсор.

## СОЗДАНИЕ КУРСОРА

Поскольку стандартные курсоры предопределены, нет необходимости в их создании. Чтобы использовать стандартный курсор, прикладная программа извлекает информацию о дескрипторе курсора, используя функцию **LoadCursor** или **LoadImage**. Дескриптор курсора (**cursor handle**) - уникальное значение типа **HCURSOR**, которое идентифицирует стандартный или пользовательский курсор.

Чтобы создать пользовательский курсор для прикладной программы, Вы должны как обычно использовать программу работы с графикой и включить курсор как ресурс в файле определения ресурса приложения. В период выполнения программы, вызовите функцию **LoadCursor**, чтобы извлечь информацию о дескрипторе курсора. Ресурсы курсора содержат данные для нескольких различных устройств отображения. Функция **LoadCursor** автоматически выбирает наиболее соответствующие данные для текущего устройства отображения. Чтобы загрузить курсор непосредственно из **.CUR** или **.ANI** файла, используйте функцию **LoadCursorFromFile**.

Вы можете также создать пользовательский курсор во время выполнения программы, используя функцию **CreateIconIndirect**, которая создает курсор, основанный на содержании структуры **ICONINFO**. Функция **GetIconInfo** заполняет эту структуру координатами острия и информацией относительно взаимодействия битовой маски (**bitmask**) и цвета.

Прикладные программы должны исполнять пользовательские курсоры как ресурсы и использовать функции **LoadCursor**, **LoadCursorFromFile** или **LoadImage**, а не курсор созданный во время выполнения программы. Использование ресурсов курсора освобождает от зависимости от устройства, упрощает определение местонахождения и дает возможность прикладным программам совместно использовать разработанные курсоры.

Функция **CreateIconFromResourceEx** дает возможность прикладной программе создать пиктограммы и курсоры, основанные на данных ресурса. **CreateIconFromResourceEx** создает курсор, основанный на двоичных данных ресурса из других исполняемых (**.EXE**) файлов или динамически подключаемых библиотек (**DLLs**). Этому должны предшествовать обращения к функции **LookupIconIdFromDirectoryEx**, также как и к нескольким функциям ресурса. Функция **LookupIconIdFromDirectoryEx** идентифицирует наиболее соответствующие данные курсора для текущего устройства отображения. Для получения дополнительной информации относительно функций ресурса, см. статью Ресурсы.

## РАСПОЛОЖЕНИЕ И ВИД КУРСОРА

**Windows** автоматически отображает курсор мыши и модифицирует его позицию на экране. Вы можете получать текущие экранные координаты курсора и перемещать курсор в любое местоположение на экране, используя, соответственно, функции **GetCursorPos** и **SetCursorPos**.

Вы можете также извлечь информацию о дескрипторе текущего курсора, используя функцию **GetCursor**, можете установить курсор, используя функцию **SetCursor**. После того, как вы вызовите **SetCursor**, вид курсора не изменяется до тех пор, пока или при перемещении мыши курсор явно устанавливается в другой курсор, или выполняется системная команда.

Когда пользователь перемещает мышь, система перерисовывает курсор в новом местоположении. Система автоматически перерисовывает курсор, на курсор связанный с окном, в котором он является указателем.

Вы можете скрыть и восстановить изображение курсора, без изменения рисунка курсора, используя функцию **ShowCursor**. Эта функция использует внутренний счетчик для определения, когда скрыть или отобразить курсор. Попытка показать курсор увеличивает счетчик; попытка скрыть курсор уменьшает счетчик. Курсор видим только тогда, если этот счетчик больше или равен нулю.

## КУРСОР КЛАССА ОКНА

Когда Вы регистрируете класс окна, используя функцию **RegisterClass**, вы можете назначить ею заданный по умолчанию курсор, известный как курсор класса. После того, как прикладная программа регистрирует класс окна, каждое окно этого класса будет иметь заданный курсор класса (**class cursor**).

Чтобы отменить курсор класса, обработайте сообщение **WM\_SETCURSOR**. Вы можете также заменить курсор класса, используя функцию **SetClassLong**. Эта функция изменяет заданные по умолчанию параметры настройки окна для всех окон данного класса. Для получения дополнительной информации, см. раздел Классы окна.

Функции курсора

*ClipCursor*

Функция ограничивает движение курсора определённой областью. Если позиция курсора находится за пределами указанной области, windows автоматически переносит его в ближайшую точку в этой области.

Существует в: Win NT, Win9x, Win32s.

Для С/С++ объявлена в winuser.h. Для Delphi в модуле windows.

Объявление:

Для С/С++

BOOL ClipCursor(

CONST RECT \*lpRect // Указатель на структуру области

);

Для Delphi

function ClipCursor(

lpRect: PRect

): BOOL; stdcall;

Функции передаётся единственный параметр в виде указателя на структуру в которой содержатся координаты области. Если всё ничтяк, то функция возвращает не ноль.

Пример использования в Delphi: ClipCursor(Rect(10,15,50,55)). В этом случае курсор ограничивается прямоугольником с координатами Х1=10, Х2=15, Y1=50, Y2=55.

CopyCursor

Функция копирования курсора. Точнее сказать, возвращает копию переданного ей курсора.

Существует в: Win NT, Win9x, Win32s.

Для С/С++ объявлена в winuser.h. Самое интересное, что для Delphi я объявления не нашёл. Но оно должно выглядеть где-то так, как приведено ниже.

Объявление:

Для С/С++

HCURSOR CopyCursor(

HCURSOR pcur // Указатель на курсор

);

Для Delphi

function CopyCursor(

pcur: HICON

): HICON; stdcall;

Функции передаётся единственный параметр в виде указателя на существующий курсор. После выполнения она вернёт копию переданного курсора.

CreateCursor

Функция создаёт новый курсор.

Существует в: Win NT, Win9x, Win32s.

Для С/С++ объявлена в winuser.h. Для Delphi в модуле windows.

Объявление:

Для С/С++

HCURSOR CreateCursor(

HINSTANCE hInst, // Указатель на экземпляр приложения

int xHotSpot, // Горизонтальная позиция горячей точки

int yHotSpot, // Вертикальная позиция горячей точки

int nWidth, // Ширина курсора

int nHeight, // Высота курсора

CONST VOID \*pvANDPlane,// Указатель на AND битовую маску

CONST VOID \*pvXORPlane // Указатель на XOR битовую маску

);

Для Delphi

function CreateCursor(

hInst: HINST; // Указатель на экземпляр приложения

xHotSpot, // Горизонтальная позиция горячей точки

yHotSpot, // Вертикальная позиция горячей точки

nWidth, // Ширина курсора

nHeight: Integer; // Высота курсора

pvANDPlaneter, // Указатель на AND битовую маску

pvXORPlane: Pointer// Указатель на XOR битовую маску

): HCURSOR; stdcall;

Горячая точка - точка которой будет происходить касание. Например, у курсора в виде прицела горячая точка должна быть в центре, а у стрелки на её конце.

DestroyCursor

Функция уничтожает курсор.

Существует в: Win NT, Win9x, Win32s.

Для С/С++ объявлена в winuser.h. Для Delphi в модуле windows.

Объявление:

Для С/С++

BOOL DestroyCursor(

HCURSOR hCursor // Указатель на уничтожаемый курсор

);

Для Delphi

function DestroyCursor(

hCursor: HICON // Указатель на уничтожаемый курсор

): BOOL; stdcall;

Функции передаётся единственный параметр в виде указателя на существующий курсор который надо уничтожить.

GetClipCursor

Функция возвращает область в которой может перемещаться курсор. Эту область можно задать с помощью функции ClipCursor.

Существует в: Win NT, Win9x, Win32s.

Для С/С++ объявлена в winuser.h. Для Delphi в модуле windows.

Объявление:

Для С/С++

BOOL GetClipCursor(

LPRECT lpRect // Структура типа Rect

);

Для Delphi

function GetClipCursor(

var lpRect: TRect // Структура типа Rect

): BOOL; stdcall;

Функции передаётся единственный параметр в виде указателя на структуру типа Rect. В неё будет записан результат, то есть размеры области курсора. Если всё ничтяк, то функция вернёт не ноль.

GetCursorPos

Функция возвращает текущее положение курсора.

Существует в: Win NT, Win9x, Win32s.

Для С/С++ объявлена в winuser.h. Для Delphi в модуле windows.

Объявление:

Для С/С++

BOOL GetCursorPos(

LPPOINT lpPoint // Структура с позицией курсора.

);

Для Delphi

function GetCursorPos(

var lpPoint: TPoint // Структура с позицией курсора.

): BOOL; stdcall;

Функции передаётся единственный параметр в виде указателя на структуру типа Point. В неё будет записан результат, т.е. текущая позиция курсора. Пример на Delphi:

var

MousePos:TPoint;

begin

GetCursorPos(MousePos);

end;

После этого, чтобы получить Х позицию курсора, нужно прочитать MousePos.х, а для y позиции нужно прочитать MousePos.y.

Если всё ничтяк, то функция вернёт не ноль.

SetCursorPos

Функция устанавливает текущее положение курсора. С её помощью ты можешь переместить указатель курсора в любое место экрана.

Существует в: Win NT, Win9x, Win32s.

Для С/С++ объявлена в winuser.h. Для Delphi в модуле windows.

Объявление:

Для С/С++

BOOL SetCursorPos(

int X, // Горизонтальная позиция

int Y // Вертикальная позиция

);

Для Delphi

function SetCursorPos(

X, // Горизонтальная позиция

Y: Integer // Вертикальная позиция

): BOOL; stdcall;

Если всё ничтяк, то функция вернёт не ноль.

ShowCursor

Эта функция позволяет показывать и прятать курсор.

Существует в: Win NT, Win9x, Win32s.

Для С/С++ объявлена в winuser.h. Для Delphi в модуле windows.

Объявление:

Для С/С++

int ShowCursor(

BOOL bShow // cursor visibility flag

);

Для Delphi

function ShowCursor(

bShow: BOOL

): Integer; stdcall;

Функции передаётся только один параметр. Если он равен true, то курсор надо показать, а если равен false, то спрятать.

Если всё хорошо, то функция вернёт не ноль.

Задание 2 присвоить приложению пиктограмму

Вот как вы это делаете в Visual Studio 2010.

Потому что это изящно, на самом деле это может быть довольно болезненно, потому что вы пытаетесь сделать что-то настолько невероятно просто, но это не прямолинейно, и есть много ошибок, о которых Visual Studio ничего не говорит. Если в любой момент вы чувствуете себя сердитым или хотите, чтобы вы опустили свои зубы в 2 на 4 и кричали, пожалуйста, сделайте это.

Gotchas:

1. Вам нужно использовать файл .ico. **Вы не можете использовать файл изображения PNG для исполняемого файла, он не будет работать. Вы должны использовать .ico.** Есть веб-утилиты, которые конвертируют изображения в файлы .ico.
2. Икон, используемый для вашего exe, будет ico с наименьшим идентификатором RESOURCE. Чтобы изменить .ico

1) Откройте **VIEW** > **RESOURCE VIEW** (в середине меню **VIEW**) или нажмите Ctrl + Shift + E, чтобы он появился.

2) В окне "Ресурсы" щелкните правой кнопкой мыши имя проекта и скажите **ADD** > **РЕСУРС...**

3) Предположим, что вы уже создали файл .ico самостоятельно, выберите **Значок** из появившегося списка, затем нажмите **ИМПОРТ**.

4) В этом диалоге \*.ico файлы не указаны, и вы **не можете** использовать обычное изображение PNG или JPG в качестве значка, поэтому измените фильтр файлов на **t20 >**, используя раскрывающийся список. Вводящий в заблуждение пользовательский интерфейс, я знаю, я знаю.

5) Если вы сейчас скомпилируете проект, он будет **автоматически** прикреплять .ico **с самым низким идентификатором** (как указано в resource.h) в качестве значка вашего .exe.

6) Если вы загружаете в проект кучу ICO файлов по какой-либо причине, убедитесь, что .ico, который вы хотите использовать Visual Studio, имеет **самый низкий** id в resource.h. Вы можете редактировать этот файл вручную без проблем

Eg.

//resource.h

#define IDI\_ICON1 102

#define IDI\_ICON2 103

Используется IDI\_ICON1

//resource.h

#define IDI\_ICON1 106

#define IDI\_ICON2 103

Теперь используется IDI\_ICON2.

Зaдание 3 Добавить акселератор

Классы [Accelerator](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-class?view=vs-2019) и [accelerator\_view](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-view-class?view=vs-2019) можно использовать для указания устройства или эмулятора, на котором будет выполняться код C++ amp. Система может иметь несколько устройств или эмуляторов, которые отличаются объемом памяти, поддержкой общей памяти, поддержкой отладки или поддержкой двойной точности. C++Ускоренный большой параллелизм (C++ amp) предоставляет интерфейсы API, которые можно использовать для проверки доступных сочетаний клавиш, по умолчанию — для нескольких вызовов parallel\_for\_each и для выполнения специальных задач отладки.

## Использование ускорителя по умолчанию

Среда C++ выполнения amp выбирает ускоритель по умолчанию, если не написать код для выбора конкретного элемента. Среда выполнения выбирает ускоритель по умолчанию следующим образом:

1. Если приложение работает в режиме отладки, ускоритель, поддерживающий отладку.
2. В противном случае — ускоритель, заданный переменной среды CPPAMP\_DEFAULT\_ACCELERATOR, если он задан.
3. В противном случае — устройство без эмуляции.
4. В противном случае устройство с наибольшим объемом доступной памяти.
5. В противном случае — устройство, не подключенное к дисплею.

Кроме того, среда выполнения задает access\_type access\_type\_auto для ускорителя по умолчанию. Это означает, что ускоритель по умолчанию использует общую память, если она поддерживается и если ее характеристики производительности (пропускная способность и задержка) совпадают с выделенной (не общей) памятью.

Чтобы определить свойства ускорителя по умолчанию, можно создать ускоритель по умолчанию и изучить его свойства. Следующий пример кода выводит на печать путь, объем памяти ускорителя, поддержку общей памяти, поддержку двойной точности и ограниченную поддержку двойной точности для ускорителя по умолчанию.

C++Копировать

void default\_properties() {

accelerator default\_acc;

std::wcout << default\_acc.device\_path << "\n";

std::wcout << default\_acc.dedicated\_memory << "\n";

std::wcout << (accs[i].supports\_cpu\_shared\_memory ?

"CPU shared memory: true" : "CPU shared memory: false") << "\n";

std::wcout << (accs[i].supports\_double\_precision ?

"double precision: true" : "double precision: false") << "\n";

std::wcout << (accs[i].supports\_limited\_double\_precision ?

"limited double precision: true" : "limited double precision: false") << "\n";

}

### Переменная среды CPPAMP\_DEFAULT\_ACCELERATOR

Можно задать переменную среды CPPAMP\_DEFAULT\_ACCELERATOR, чтобы указать значение accelerator::device\_path ускорителя по умолчанию. Путь зависит от оборудования. Следующий код использует accelerator::get\_all функцию для получения списка доступных сочетаний клавиш, а затем отображает путь и характеристики каждого ускорителя.

C++Копировать

void list\_all\_accelerators()

{

std::vector<accelerator> accs = accelerator::get\_all();

for (int i = 0; i <accs.size(); i++) {

std::wcout << accs[i].device\_path << "\n";

std::wcout << accs[i].dedicated\_memory << "\n";

std::wcout << (accs[i].supports\_cpu\_shared\_memory ?

"CPU shared memory: true" : "CPU shared memory: false") << "\n";

std::wcout << (accs[i].supports\_double\_precision ?

"double precision: true" : "double precision: false") << "\n";

std::wcout << (accs[i].supports\_limited\_double\_precision ?

"limited double precision: true" : "limited double precision: false") << "\n";

}

}

## Выбор сочетания клавиш

Чтобы выбрать ускоритель, используйте accelerator::get\_all метод для получения списка доступных сочетаний клавиш, а затем выберите один из его свойств. В этом примере показано, как выбрать ускоритель с наибольшим объемом памяти:

C++Копировать

void pick\_with\_most\_memory()

{

std::vector<accelerator> accs = accelerator::get\_all();

accelerator acc\_chosen = accs[0];

for (int i = 0; i <accs.size(); i++) {

if (accs[i].dedicated\_memory> acc\_chosen.dedicated\_memory) {

acc\_chosen = accs[i];

}

}

std::wcout << "The accelerator with the most memory is "

<< acc\_chosen.device\_path << "\n"

<< acc\_chosen.dedicated\_memory << ".\n";

}

**Примечание**

Один из ускорителей, возвращаемых accelerator::get\_all , — это ускоритель ЦП. Невозможно выполнить код в ускорителе ЦП. Чтобы отфильтровать ускоритель ЦП, Сравните значение свойства **[device\_path](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-class?view=vs-2019" \l "device_path)** ускорителя, возвращаемое accelerator::get\_all , со значением [**ускорителя:: cpu\_accelerator**](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-class?view=vs-2019#cpu_accelerator). Дополнительные сведения см. в разделе "специальные ускорители" этой статьи.

## Общая память

Общая память — это память, доступ к которой может осуществляться как ЦП, так и ускорителем. Использование общей памяти устраняет или значительно сокращает затраты на копирование данных между ЦП и ускорителем. Несмотря на то, что память является общей, она недоступна одновременно с ЦП и ускорителем, и это приводит к неопределенному поведению. Свойство ускорителя [supports\_cpu\_shared\_memory](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-class?view=vs-2019" \l "supports_cpu_shared_memory) возвращает **значение true** , если ускоритель поддерживает общую память, а свойство [default\_cpu\_access\_type](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-class?view=vs-2019" \l "default_cpu_access_type) возвращает [access\_type](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/concurrency-namespace-enums-amp?view=vs-2019" \l "access_type) по умолчанию для памяти, выделенной дляaccelerator— например, **массивы**, acceleratorсвязанные с объектами accelerator, или array\_view доступ к которым осуществляется в.

Среда C++ выполнения amp автоматически выбирает оптимальное значение access\_type по умолчанию для каждого accelerator, но характеристики производительности (пропускная способность и задержка) для общей памяти могут быть хуже, чем в выделенной (без общего) памяти ускорителя, если чтение из ЦП, запись из ЦП или и то, и другое. Если общая память работает, а также выделенная память для чтения и записи из ЦП, среда выполнения по умолчанию access\_type\_read\_writeпринимает значение; в противном случае среда выполнения выбирает access\_typeболее консервативное значение по умолчанию и позволяет приложению переопределить его, если доступ к памяти шаблоны вычислительных ядер имеют другое access\_typeпреимущество.

В следующем примере кода показано, как определить, поддерживает ли ускоритель по умолчанию общую память, а затем переопределить тип доступа по умолчанию и accelerator\_view создать из него.

C++Копировать

#include <amp.h>

#include <iostream>

using namespace Concurrency;

int main()

{

accelerator acc = accelerator(accelerator::default\_accelerator);

// Early out if the default accelerator doesn’t support shared memory.

if (!acc.supports\_cpu\_shared\_memory)

{

std::cout << "The default accelerator does not support shared memory" << std::endl;

return 1;

}

// Override the default CPU access type.

acc.set\_default\_cpu\_access\_type(access\_type\_read\_write);

// Create an accelerator\_view from the default accelerator. The

// accelerator\_view reflects the default\_cpu\_access\_type of the

// accelerator it’s associated with.

accelerator\_view acc\_v = acc.default\_view;

}

Всегда отражает объект default\_cpu\_access\_type ,accelerator с которым связан объект, и не access\_typeпредоставляет интерфейс для переопределения или изменения. accelerator\_view

## Изменение ускорителя по умолчанию

Можно изменить ускоритель по умолчанию, вызвав accelerator::set\_default метод. Вы можете изменить ускоритель по умолчанию только один раз для каждого выполнения приложения. его необходимо изменить перед выполнением кода на GPU. Все последующие вызовы функций для изменения ускорителя возвращают **значение false**. Если вы хотите использовать другой ускоритель в вызове parallel\_for\_each, ознакомьтесь с разделом "использование нескольких ускорителей" этой статьи. В следующем примере кода в качестве ускорителя по умолчанию устанавливается один, который не эмулируется, не подключается к дисплею и поддерживает двойную точность.

C++Копировать

bool pick\_accelerator()

{

std::vector<accelerator> accs = accelerator::get\_all();

accelerator chosen\_one;

auto result = std::find\_if(accs.begin(), accs.end(),

[] (const accelerator& acc) {

return !acc.is\_emulated &&

acc.supports\_double\_precision &&

!acc.has\_display;

});

if (result != accs.end()) {

chosen\_one = \*(result);

}

std::wcout <<chosen\_one.description <<std::endl;

bool success = accelerator::set\_default(chosen\_one.device\_path);

return success;

}

## Использование нескольких ускорителей

Существует два способа использования нескольких ускорителей в приложении:

* Объекты можно передавать accelerator\_view в вызовы метода [parallel\_for\_each](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/concurrency-namespace-functions-amp?view=vs-2019" \l "parallel_for_each) .
* Объект **Array** можно создать с помощью определенного accelerator\_view объекта. Среда выполнения accelerator\_view C + amp берет объект из захваченного объекта **массива** в лямбда-выражении.

## Специальные ускорители

В качестве свойств accelerator класса доступны пути к устройствам трех специальных ускорителей:

* [ускоритель::D элемент данных irect3d\_ref](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-class?view=vs-2019#direct3d_ref): Это однопотоковое сочетание клавиш использует программное обеспечение ЦП для эмуляции универсальной графической карты. Он используется по умолчанию для отладки, но он не полезен в рабочей среде, так как он медленнее, чем ускорители оборудования. Кроме того, он доступен только в пакете SDK DirectX и Windows SDK, и вряд ли будет установлен на компьютерах клиентов. Дополнительные сведения см. в разделе [Отладка кода GPU](https://docs.microsoft.com/ru-ru/visualstudio/debugger/debugging-gpu-code).
* [ускоритель::D элемент данных irect3d\_warp](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-class?view=vs-2019#direct3d_warp): Этот ускоритель предоставляет резервное решение для запуска кода C++ amp на многоядерных ЦП, использующих расширения Streaming SIMD (SSE).
* [элемент данных ускорителя:: cpu\_accelerator](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/accelerator-class?view=vs-2019#cpu_accelerator): Этот ускоритель можно использовать для настройки промежуточных массивов. Он не может C++ выполнить код amp. Дополнительные сведения см. в разделе [промежуточные массивы C++ в](https://blogs.msdn.microsoft.com/nativeconcurrency/2011/11/09/staging-arrays-in-c-amp/) элементе управления amp Post в блоге по параллельному программированию в машинном коде.

## Взаимодействие

Среда C++ выполнения amp поддерживает взаимодействие между accelerator\_view классом и интерфейсом Direct3D [ID3D11Device](https://docs.microsoft.com/ru-ru/windows/win32/api/d3d11/nn-d3d11-id3d11device). Метод [create\_accelerator\_view](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/concurrency-direct3d-namespace-functions-amp?view=vs-2019#create_accelerator_view) IUnknown принимаетaccelerator\_view интерфейс и возвращает объект. Метод [get\_device](https://docs.microsoft.com/ru-ru/cpp/parallel/amp/reference/concurrency-direct3d-namespace-functions-amp?view=vs-2019#get_device) accelerator\_view принимаетIUnknown объект и возвращает интерфейс.