



Нижегородский государственный университет им. Н.И. Лобачевского

***Разработка мультимедийных приложений
с использованием библиотек OpenCV и IPP***

Лекция
Обзор библиотек OpenCV и Intel IPP

При поддержке компании Intel

Золотых Н.Ю., Половинкин А.Н.

Содержание

- ❑ Обзор библиотеки OpenCV:
 - TODO
- ❑ Обзор библиотеки IPP:
 - структура библиотеки и описание назначения ключевых модулей
 - области применения
 - типы данных и способы именования функций
 - вопросы производительности
 - пример использования библиотеки в среде MS Visual Studio

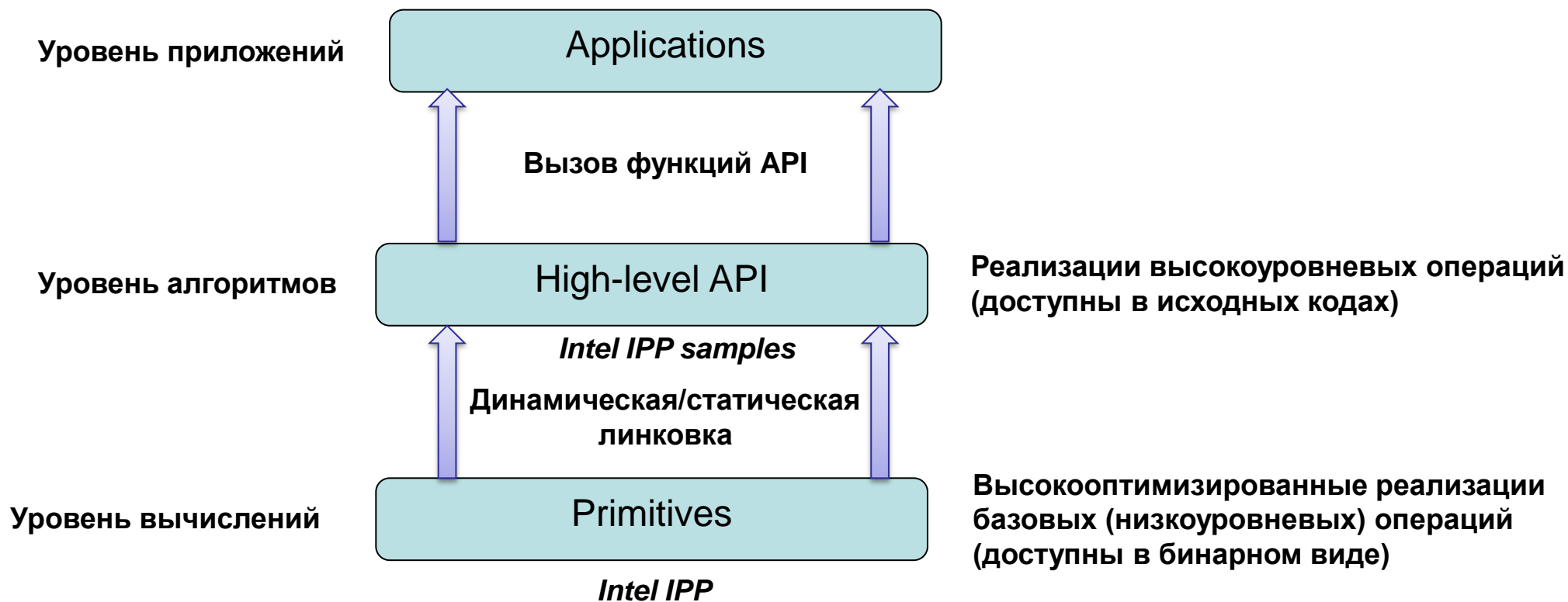


Области применения библиотеки

- ❑ Intel IPP (Integrated Performance Primitives) - это набор кроссплатформенных библиотек, оптимизированных под Intel CPU, содержащих большое количество высокопроизводительных функций, которые могут быть использованы в следующих областях:
 - Видео-кодеки: H.264, MPEG2, MPEG4, VC-1
 - Audio-кодеки: AAC, AC3, MP3
 - Сжатие изображений (кодеки JPEG, JPEG2000, JPEGXR)
 - Обработка изображений
 - Обработка сигналов
 - Сжатие естественной речи (кодеки AMR, AMR-WBE, G.711, G.722, G.723, G.726, G.728, G.729, GSM-AMR, GSM-MFR)
 - Криптографические приложения



Структура библиотеки



Типы данных

Type	Usual C type	Intel IPP Type
8u	unsigned char	lpp8u
8s	signed char	lpp8s
16u	unsigned short	lpp16u
16s	signed short	lpp16s
16sc	complex short	lpp16sc
32u	unsigned int	lpp32u
32s	signed int	lpp32s
32f	float	lpp32f
32fc	complex float	lpp32fc
64s	__int64 (long long)	lpp64s
64f	double	lpp64f
64fc	complex double	lpp64fc



Типы данных и именование функций (1)

```
ipp<data-domain><name>[_<datatype>][_<descriptor>](<parameters>) ;
```

□ **<data-domain>** (тип обрабатываемых данных):

- s (signal) – одномерный массив
- i (image, video) – двумерный массив пикселей
- m (matrice) – матрицы или вектора
- r – исходные данные для функций реалистичного рендеринга изображений и обработки 3D данных (представление данных зависит от используемой функции)

Примеры: ipps, ipri, iprm, ippr



Типы данных и именование функций (2)

```
ipp<data-domain><name>[_<datatype>] [_<descriptor>] (<parameters>) ;
```

□ **<name>** = <operation>[_modifier] (имя функции)

- **<operation>** - название выполняемой операции
- **modifier** (дополнительный параметр) – дополнительная спецификация функции

Примеры: Cору, Malloc, DCTFwd_CToC (прямое дискретное косинус-преобразование, в котором входные и выходные значения являются комплексными)

Типы данных и именование функций (3)

`ipp<data-domain><name>[_<datatype>][_<descriptor>](<parameters>);`

□ **<datatype>** = <bit_depth><bit_interpretation> (тип обрабатываемых данных)

- **<bit_depth>** = 1|8|16|32|64 (размер элемента данных (в битах))
- **<bit_interpratation>** = <u|s|f>[c]
 - u – целые числа без знака
 - s – целые числа со знаком
 - f – вещественные числа
 - c – комплексные числа

В случае, если у функции есть несколько аргументов разных типов (или тип выходных данных отличается от типа входных данных) формат типа обрабатываемых данных:

`<datatype> = <src1Datatype>[<src2Datatype>][<dstDatatype>]`



Типы данных и именование функций (4)

```
ipp<data-domain><name>[_<datatype>] [_<descriptor>] (<parameters>) ;
```

- **<descriptor>** - описание дополнительных особенностей данных, обрабатываемых функцией

Примеры:

M – используется маска, определяющая пиксели изображения, которые будут обработаны функцией

R – функция обрабатывает определенную область изображения (ROI – region of interest)

Типы данных и именование функций (5)

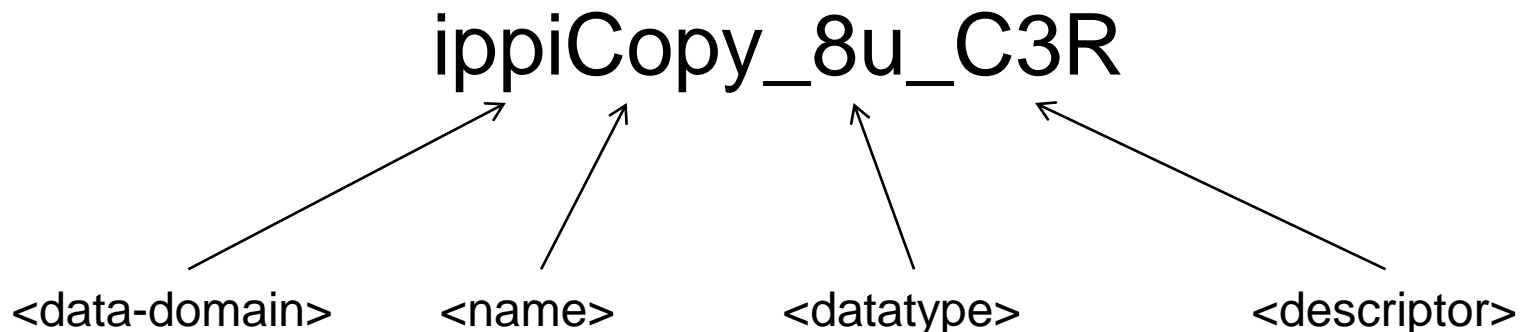
```
ipp<data-domain><name>[_<datatype>][_<descriptor>](<parameters>) ;
```

□ **<parameters>** - параметры функции

Параметры функции указываются в следующем порядке:

- Входные аргументы
- Выходные аргументы
- Дополнительные аргументы

Типы данных и именование функций (6)



Копирует (**Copy**) изображение (**ippi**), элементами которого являются 8-битные целые числа без знака (**8u**). Изображение является трехканальным (**C3**), копирование осуществляется из заданной области изображения (**R**).

Выделение и освобождение памяти

- ❑ `Ipp<datatype>* ippMalloc_<datatype>(int len)`

Выделяет память под одномерный массив из **len** элементов типа **<datatype>**, выровненный по 32 байт.

- ❑ `ippFree(void* ptr)`

Освобождает память, выделенную при помощи **ippMalloc**

- ❑ `Ipp<datatype>* ippiMalloc_<mod>(int widthPixels, int heightPixels, int* pStepBytes)`

Выделяет память под изображение шириной в **widthPixels**, высотой в **heightPixels**. Каждая строка изображения выровнена по 32 байта, расстояние между последовательными строками возвращается в массиве **pStepBytes**. **<mod>** описывает тип изображения (например, **8u_C3** – трехканальное изображение, интенсивность каждого пикселя описывается unsigned char)

- ❑ `ippiFree(void* ptr)`

Освобождает память, выделенную при помощи **ippiMalloc**



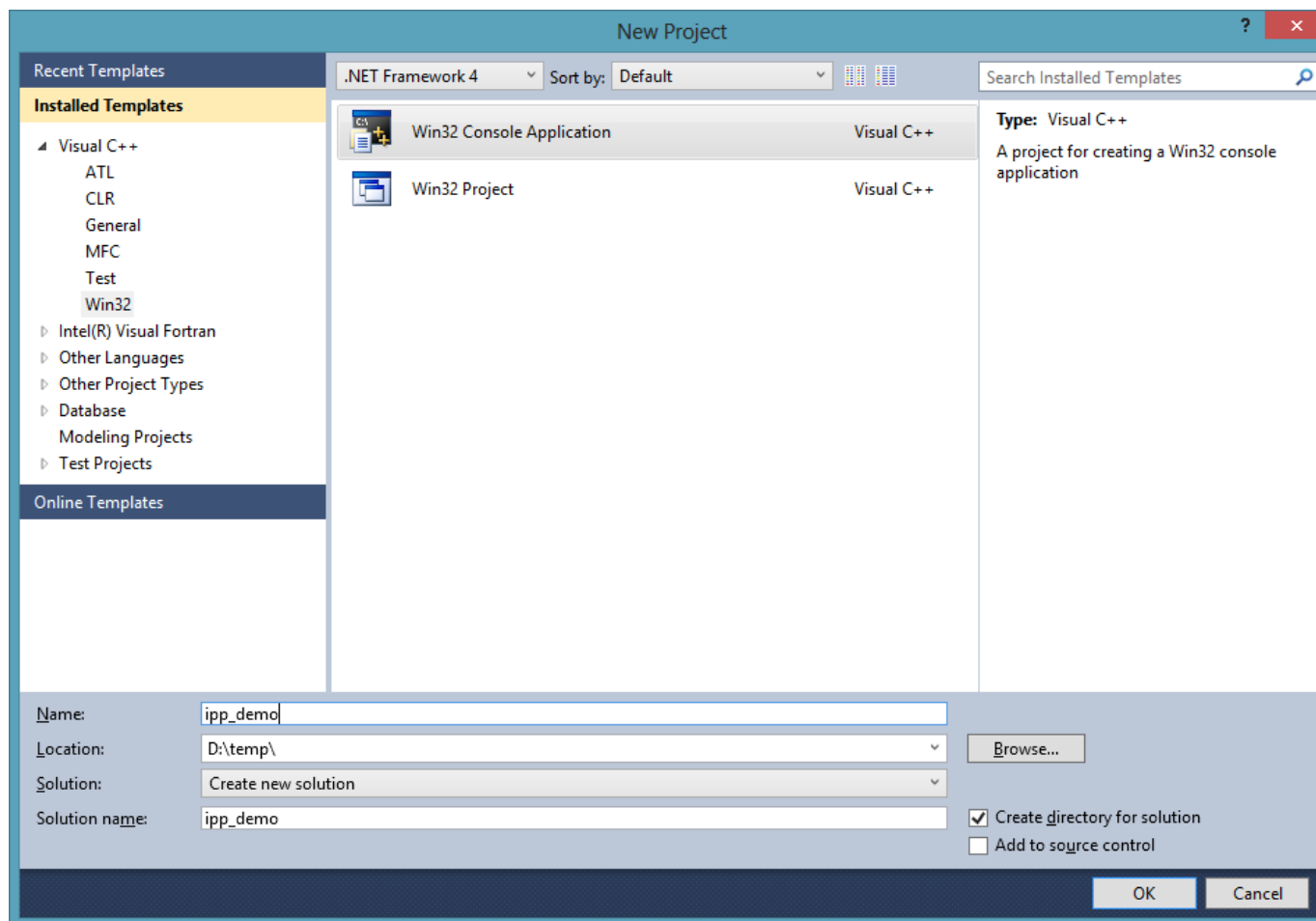
Вопросы производительности

- Intel IPP содержит ряд функций, обеспечивающих выбор реализации, оптимизированной под конкретную архитектуру:
 - `IppStatus ippStaticInit(void)` - определяет тип используемого процессора и выбирает наиболее подходящий для данного процессора статический код. Данная функция применима лишь при статической линковке.
 - `IppStatus ippInit(void)` - определяет тип используемого процессора и выбирает наиболее подходящий для данного процессора статический код.



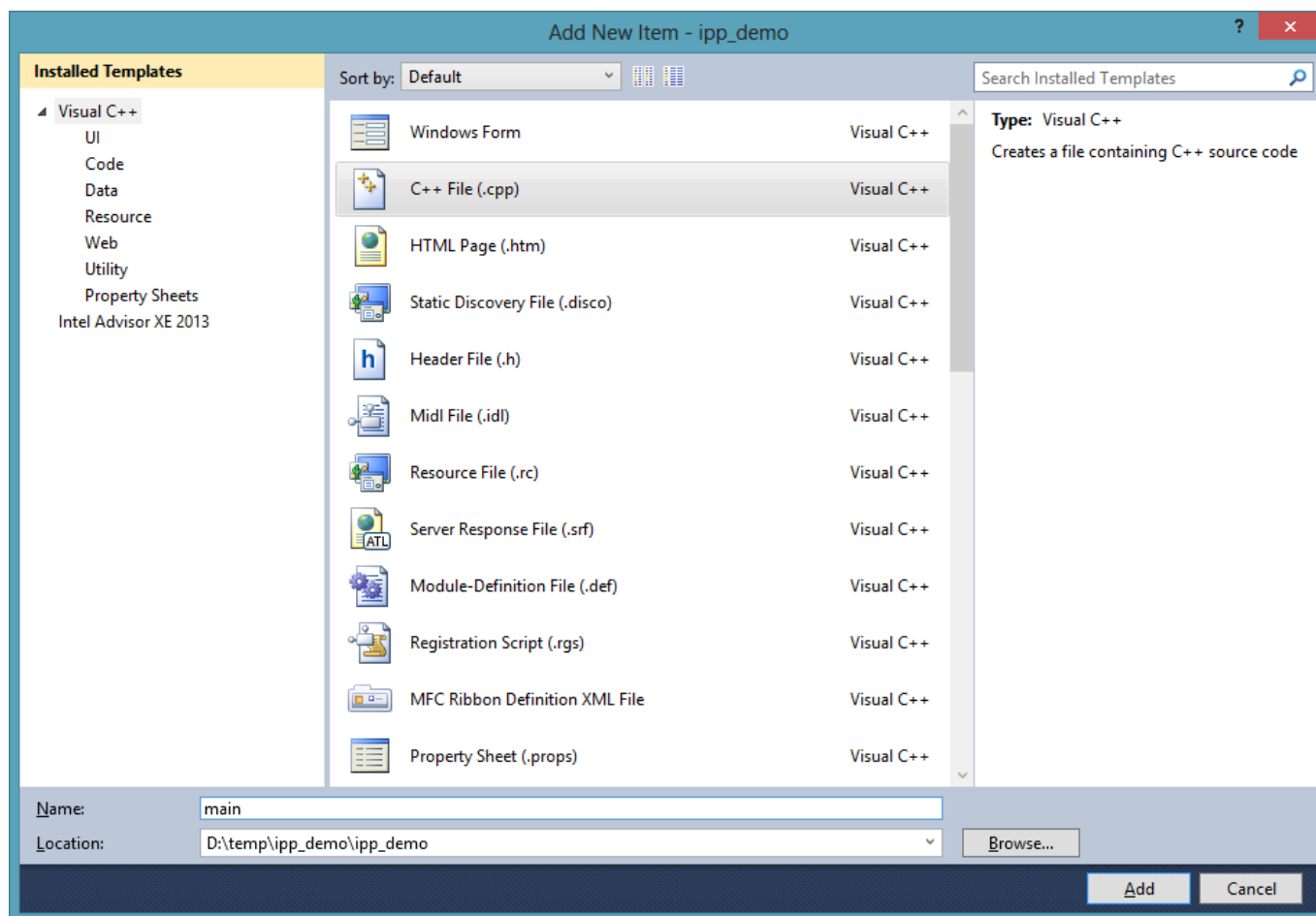
Создание приложения в MS Visual Studio 2010

□ File->New->Project



Создание приложения в MS Visual Studio 2010

- ❑ Project->Add new item->Visual C++->.cpp file



Создание приложения в MS Visual Studio 2010

```
#include "ipp.h"
#include <stdio.h>

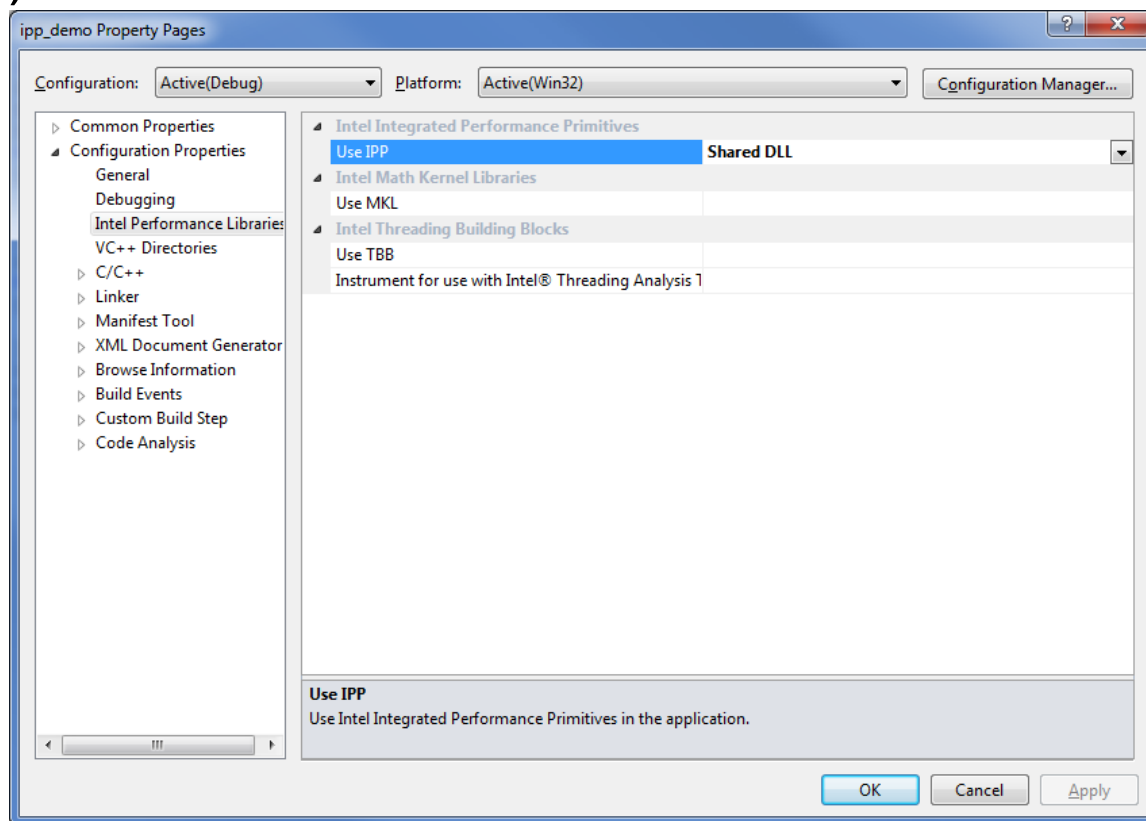
void main()
{
    int n = 5;
    Ipp32f* a = ippsMalloc_32f(n);
    Ipp32f* b = ippsMalloc_32f(n);
    Ipp32f* c = ippsMalloc_32f(n);
    for (int i = 0; i < n; i++)
    {
        a[i] = (float)i;
        b[i] = (float)(10 - i);
    }
    ippsAdd_32f(a, b, c, n);

    for (int i = 0; i < n; i++)
    {
        printf("%.3f + %.3f\n", a[i], b[i], c[i]);
    }
    ippsFree(a);
    ippsFree(b);
    ippsFree(c);
}
```



Создание приложения в MS Visual Studio 2010

Configuration properties->Intel Performance Libraries->Use IPP->(Shared DLL, Single-threaded static library, Multi-threaded static library)



Вопросы

?

