

# Информатика. Семинар №11

Json, многопоточность в C++11,  
звук в sfml

# Вопрос №1

```
#include<iostream>
using namespace std;

class Base {};

class Derived: public Base {};

int main()
{
    Base *bp = new Derived;
    Derived *dp = new Base;
}
```

- Ok
- CE in line «Base\* bp = new Derived;»
- CE in line «Derived\* dp = new Base;»
- Runtime Error

# Вопрос №1

```
#include<iostream>
using namespace std;

class Base {};

class Derived: public Base {};

int main()
{
    Base *bp = new Derived;
    Derived *dp = new Base;
}
```

- Ok
- CE in line «Base\* bp = new Derived;»
- CE in line «Derived\* dp = new Base;»
- Runtime Error

# Вопрос №2

```
class Base
{
public:
    void show()
    {
        cout<<" In Base ";
    }
};

class Derived: public Base
{
public:
    int x;
    void show()
    {
        cout<<"In Derived ";
    }
    Derived()
    {
        x = 10;
    }
};

int main(void)
{
    Base *bp, b;
    Derived d;
    bp = &d;
    bp->show();
    cout << bp->x;
    return 0;
}
```

- CE in line «bp->show();»
- CE in line «cout << bp->x;»
- In Base 10
- In Derived 10

# Вопрос №2

```
class Base
{
public:
    void show()
    {
        cout<<" In Base ";
    }
};

class Derived: public Base
{
public:
    int x;
    void show()
    {
        cout<<"In Derived ";
    }
    Derived()
    {
        x = 10;
    }
};

int main(void)
{
    Base *bp, b;
    Derived d;
    bp = &d;
    bp->show();
    cout << bp->x;
    return 0;
}
```

- CE in line «bp->show();»
- CE in line «cout << bp->x;»
- In Base 10
- In Derived 10

# Вопрос №3

```
#include<iostream>
using namespace std;

class Base
{
public:
    int fun() { cout << "Base::fun() called"; }
    int fun(int i) { cout << "Base::fun(int i) called"; }
};

class Derived: public Base
{
public:
    int fun() { cout << "Derived::fun() called"; }
};

int main()
{
    Derived d;
    d.fun(5);
    return 0;
}
```

- Base::fun() called
- Base::fun(int i) called
- Derived::fun() called
- Compilation error

# Вопрос №3

```
#include<iostream>
using namespace std;

class Base
{
public:
    int fun() { cout << "Base::fun() called"; }
    int fun(int i) { cout << "Base::fun(int i) called"; }
};

class Derived: public Base
{
public:
    int fun() { cout << "Derived::fun() called"; }
};

int main()
{
    Derived d;
    d.fun(5);
    return 0;
}
```

- Base::fun() called
- Base::fun(int i) called
- Derived::fun() called
- **Compilation error**

If a derived class writes its own method, then all functions of base class with same name become hidden, even if signatures of base class functions are different. In the above question, when fun() is rewritten in Derived, it hides both fun() and fun(int) of base class.

# Вопрос №4

```
#include <iostream>
using namespace std;

template <typename T>
void fun(const T&x)
{
    static int count = 0;
    cout << "x = " << x << " count = " << count << endl;
    ++count;
    return;
}

int main()
{
    fun<int>(1);
    cout << endl;
    fun<int>(1);
    cout << endl;
    fun<double>(1.1);
    cout << endl;
    return 0;
}
```

A

```
x = 1 count = 0
x = 1 count = 1
x = 1.1 count = 0
```

B

```
x = 1 count = 0
x = 1 count = 0
x = 1.1 count = 0
```

C

```
x = 1 count = 0
x = 1 count = 1
x = 1.1 count = 2
```

D

Compiler Error



# Вопрос №4



```
x = 1 count = 0  
  
x = 1 count = 1  
  
x = 1.1 count = 0
```

```
#include <iostream>  
using namespace std;  
  
template <typename T>  
void fun(const T&x)  
{  
    static int count = 0;  
    cout << "x = " << x << " count = " << count << endl;  
    ++count;  
    return;  
}  
  
int main()  
{  
    fun<int>(1);  
    cout << endl;  
    fun<int>(1);  
    cout << endl;  
    fun<double>(1.1);  
    cout << endl;  
    return 0;  
}
```

Compiler creates a new instance of a template function for every data type. So compiler creates two functions in the above example, one for int and other for double. Every instance has its own copy of static variable. The int instance of function is called twice, so count is incremented for the second call.

# Вопрос №5

```
#include<iostream>
#include<stdlib.h>
using namespace std;

template<class T, class U>
class A {
    T x;
    U y;
    static int count;
};

int main() {
    A<char, char> a;
    A<int, int> b;
    cout << sizeof(a) << endl;
    cout << sizeof(b) << endl;
    return 0;
}
```

A

6  
12

B

2  
8

C

Compiler Error: There can not be more than one template arguments.

D

8  
8

# Вопрос №5

```
#include<iostream>
#include<stdlib.h>
using namespace std;
```

```
template<class T, class U>
class A {
    T x;
    U y;
    static int count;
};
```

```
int main() {
    A<char, char> a;
    A<int, int> b;
    cout << sizeof(a) << endl;
    cout << sizeof(b) << endl;
    return 0;
}
```

Compiler Error: There can not be more than one template arguments.

A

6  
12



2  
8

C

D

8  
8

# Вопрос №6

A

Compiler error, template argument must be a data type.

B

10  
1

C

10000  
256

D

1  
1

```
#include <iostream>
using namespace std;

template <class T, int max>
int arrMin(T arr[], int n)
{
    int m = max;
    for (int i = 0; i < n; i++)
        if (arr[i] < m)
            m = arr[i];

    return m;
}

int main()
{
    int arr1[] = {10, 20, 15, 12};
    int n1 = sizeof(arr1)/sizeof(arr1[0]);

    char arr2[] = {1, 2, 3};
    int n2 = sizeof(arr2)/sizeof(arr2[0]);

    cout << arrMin<int, 10000>(arr1, n1) << endl;
    cout << arrMin<char, 256>(arr2, n2);
    return 0;
}
```

# Вопрос №6



Compiler error, template argument must be a data type.



10

1



10000

256



1

1

```
#include <iostream>
using namespace std;

template <class T, int max>
int arrMin(T arr[], int n)
{
    int m = max;
    for (int i = 0; i < n; i++)
        if (arr[i] < m)
            m = arr[i];

    return m;
}

int main()
{
    int arr1[] = {10, 20, 15, 12};
    int n1 = sizeof(arr1)/sizeof(arr1[0]);

    char arr2[] = {1, 2, 3};
    int n2 = sizeof(arr2)/sizeof(arr2[0]);

    cout << arrMin<int, 10000>(arr1, n1) << endl;
    cout << arrMin<char, 256>(arr2, n2);
    return 0;
}
```

# Вопрос №7

```
#include <iostream>
using namespace std;

template <class T>
T max (T &a, T &b)
{
    return (a > b)? a : b;
}

template <>
int max <int> (int &a, int &b)
{
    cout << "Called ";
    return (a > b)? a : b;
}

int main ()
{
    int a = 10, b = 20;
    cout << max <int> (a, b);
}
```

A	20
B	Called 20
C	Compiler Error

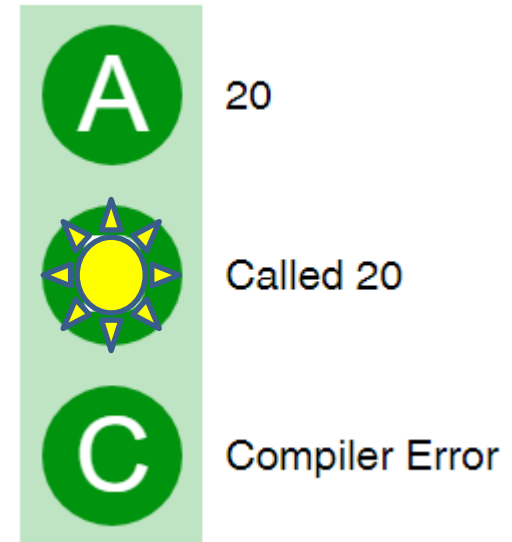
# Вопрос №7

```
#include <iostream>
using namespace std;

template <class T>
T max (T &a, T &b)
{
    return (a > b)? a : b;
}

template <>
int max <int> (int &a, int &b)
{
    cout << "Called ";
    return (a > b)? a : b;
}

int main ()
{
    int a = 10, b = 20;
    cout << max <int> (a, b);
}
```



# Вопрос №8

```
#include <iostream>
using namespace std;

template<int n> struct funStruct
{
    static const int val = 2*funStruct<n-1>::val;
};

template<> struct funStruct<0>
{
    static const int val = 1 ;
};

int main()
{
    cout << funStruct<10>::val << endl;
    return 0;
}
```



# Вопрос №8

```
#include <iostream>
using namespace std;

template<int n> struct funStruct
{
    static const int val = 2*funStruct<n-1>::val;
};

template<> struct funStruct<0>
{
    static const int val = 1 ;
};

int main()
{
    cout << funStruct<10>::val << endl;
    return 0;
}
```

# Вопрос №9

```
#include<iostream>
using namespace std;

int x[100];
int main()
{
    cout << x[99] << endl;
}
```

A

Unpredictable

B

Runtime error

C

0





D

99

# Вопрос №9

```
#include<iostream>
using namespace std;

int x[100];
int main()
{
    cout << x[99] << endl;
}
```

-  A Unpredictable
-  B Runtime error
-  C 0
-  D 99

The correct answer is c. In C++ all the uninitialized global variables are initialized to 0.

# Что такое конфигурационный файл?

- Нужен для параметризации работы программы, чтобы задать её настройки без пересборки программы
- Для этого используют, например, \*.ini файлы, либо \*.xml, либо \*.json

# Json

- <https://en.wikipedia.org/wiki/JSON>

- числа (вещественные+целые)
- строки
- boolean
- массивы []
- объекты {} (похожи на std::map)
- null

```
{  
  "firstName": "John",  
  "lastName": "Smith",  
  "isAlive": true,  
  "age": 25,  
  "address": {  
    "streetAddress": "21 2nd Street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": "10021-3100"  
  },  
  "phoneNumbers": [  
    {  
      "type": "home",  
      "number": "212 555-1234"  
    },  
    {  
      "type": "office",  
      "number": "646 555-4567"  
    },  
    {  
      "type": "mobile",  
      "number": "123 456-7890"  
    }  
  ],  
  "children": [],  
  "spouse": null  
}
```

# Библиотека jsoncpp

```
struct Person
{
    std::string firstName;
    std::string lastName;
    int age;

    struct Address
    {
        std::string street;
        std::string city;
        std::string state;
        std::string postalCode;
    } address;

    struct PhoneNumber
    {
        std::string type;
        std::string number;
    };

    std::vector<PhoneNumber> phoneNumbers;

    struct Gender
    {
        std::string type;
    } gender;

    void Parse(const std::string& fileName);
};
```

```
void Person::Parse(const std::string& fileName)
{
    std::ifstream file(fileName, std::ifstream::binary);

    if (!file.is_open())
        return;

    Json::Value root;
    file >> root;

    firstName = root["firstName"].asString();
    lastName = root["lastName"].asString();
    age = root["age"].asInt();

    auto addressValue = root["address"];
    address.city = addressValue["city"].asString();
    address.postalCode = addressValue["postalCode"].asString();
    address.state = addressValue["state"].asString();
    address.street = addressValue["streetAddress"].asString();

    auto numberValues = root["phoneNumber"];
    for (const auto& number : numberValues)
    {
        phoneNumbers.push_back({ number["type"].asString(), number["number"].asString() });
    }

    gender.type = root["gender"].get("type", "male").asString();
}
```

# Упражнение 0

- Написать заготовку config-файла + считывание для своего проекта (15 мин).



# Многопоточность в C++11

```
#include <iostream>
#include <thread>
#include <mutex>

int main() {
    int x = 0;

    std::thread t0([&]() {
        for (int i = 0; i < 1000000; ++i) x++;
    });

    std::thread t1([&]() {
        for (int i = 0; i < 1000000; ++i) x--;
    });

    t0.join(); t1.join();

    std::cout << x << std::endl;
    return 0;
}
```

P.S. Форматирование кода определяется нехваткой места на слайде

# join vs detach

- <https://habrahabr.ru/post/182610/>
- ***join*** дожидается завершения работы нити
- ***detach*** позволяет отсоединить поток от объекта, иными словами, сделать его фоновым (полезно, если мы запускаем нить в отдельной ф-и, а нить создана на стеке, т.е. должна быть удалена при выходе из неё)
- Для нити обязательно надо вызвать либо `join`, либо `detach`

# std::mutex

```
int main() {  
    int x = 0;  
    std::mutex m;  
  
    std::thread t0([&]() {  
        for (int i = 0; i < 1000000; ++i) {  
            m.lock(); x++; m.unlock();  
        }  
    });  
  
    std::thread t1([&]() {  
        for (int i = 0; i < 1000000; ++i) {  
            m.lock(); x--; m.unlock();  
        }  
    });  
  
    t0.join(); t1.join();  
  
    std::cout << x << std::endl;  
    return 0;  
}
```

# Самый простой пример dead-lock`а

```
int main(){
    std::mutex m0, m1;

    std::thread t0([&]() {
        for (int i = 0; i < 10000; ++i) {
            m0.lock(); m1.lock(); m0.unlock(); m1.unlock();
        }
    });

    std::thread t1([&]() {
        for (int i = 0; i < 10000; ++i) {
            m1.lock(); m0.lock(); m0.unlock(); m1.unlock();
        }
    });

    t0.join(); t1.join();
    std::cout << "Done";
    return 0;
}
```

# std::lock\_guard

```
int main(){
    int x=0;
    std::mutex m;

    std::thread t0([&]() {
        for (int i=0; i<1000000; ++i) {
            std::lock_guard<std::mutex> g(m);
            x++;
        }
    });

    std::thread t1([&]() {
        for (int i=0; i<1000000; ++i) {
            std::lock_guard<std::mutex> g(m);
            x--;
        }
    });

    t0.join(); t1.join();
    std::cout<<x<<std::endl;
    return 0;
}
```

# А если измерить время работы?

```
#include <iostream>
#include <thread>
#include <mutex>
#include <chrono>

int main() {
    int x = 0; const int N = 100000000; std::mutex m;

    typedef std::chrono::high_resolution_clock clock;
    clock::time_point start = clock::now();

    std::thread t0([&]() {
        for (int i = 0; i < N; ++i) { std::lock_guard<std::mutex> g(m); x++; }
    });

    std::thread t1([&]() {
        for (int i = 0; i < N; ++i) { std::lock_guard<std::mutex> g(m); x--; }
    });

    t0.join(); t1.join();

    std::chrono::duration<double> duration = clock::now() - start;
    std::cout << x << " " << duration.count() << "s" << std::endl;
    return 0;
}
```

Код с синхронизацией с помощью мьютексов работает в десятки-сотни раз дольше

# std::atomic

Для типов с тривиальным конструктором копирования (все скаляры / в случае класса сами явно не переопределяли его) можно использовать std::atomic. Может работать в разы-десятки раз быстрее std::mutex.

```
#include <atomic>

int main() {
    std::atomic<int> x = 0; const int N = 100000000;

    typedef std::chrono::high_resolution_clock clock;
    clock::time_point start = clock::now();

    std::thread t0([&]() {
        for (int i = 0; i < N; ++i) x++;
    });
```

# Lock-free programming\*: про что это? (compare and swap)

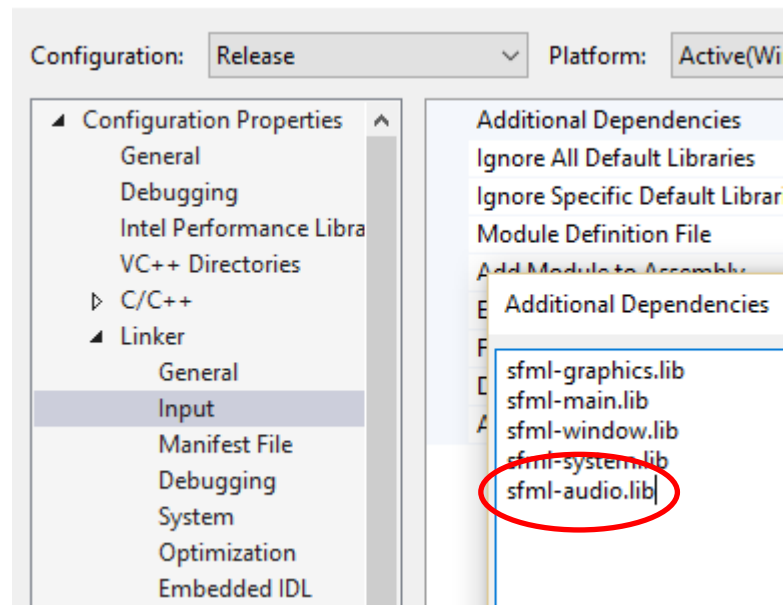
- <http://preshing.com/20120612/an-introduction-to-lock-free-programming/>
- <http://users.livejournal.com/foreseer/34284.html>
- <http://kukuruku.co/hub/cpp/lock-free-data-structures-the-evolution-of-a-stack>
- <https://habrahabr.ru/post/195770/>

P.S. Если вы в проекте собираете данные из нескольких нитей в одну структуру, то желательно разобраться и протестировать производительность в сравнении с mutex-based structures.



# Работа со звуком в sfml

- Говорим, что необходимо ссылаться с соответствующей библиотекой (для Debug и Release отдельно).



# Работа со звуком в sfml

- <http://www.sfml-dev.org/tutorials/2.3/audio-sounds.php>

```
#include <SFML\Audio.hpp>

int main()
{
    sf::Music music;
    if (!music.openFromFile("music.ogg"))
        return -1; // error
    music.play();

    while (window.isOpen())
    {
        // ...
    }
}
```

P.S. SFML supports the audio file formats WAV, OGG/Vorbis and FLAC. Due to licensing issues MP3 is **not** supported.

P.P.S. sf::Music и sf::Sound проигрываются в отдельных потоках + могут автоматически микшироваться

# Упражнение 1

- По циклу должно проигрываться несколько файлов, заданных в массиве строк:

```
std::vector<std::string> files({"file0.wav", "file1.wav", "file2.wav"});
```

```
class MusicPlayer {  
public:  
    void SetMusic(const std::vector<std::string>& musicFiles) {  
        // TODO  
    }  
    void Play() {  
        // TODO: проигрывание музыки должно происходить в отдельном потоке  
    }  
};
```

# Звуки

- Звуки – небольшие файлы, которые вызываются части и помещаются в оперативную память

```
#include <SFML/Audio.hpp>

int main()
{
    sf::SoundBuffer buffer;
    if (!buffer.loadFromFile("sound.wav"))
        return -1;

    ...

    return 0;
}
```

# Звуки

- Можно генерировать программно

```
std::vector<sf::Int16> samples = ...;  
buffer.loadFromSamples(&samples[0], samples.size(), 2, 44100);
```

Since ***loadFromSamples*** loads a raw array of samples rather than an audio file, it requires additional arguments in order to have a complete description of the sound. The first one (third argument) is the number of channels; 1 channel defines a mono sound, 2 channels define a stereo sound, etc. The second additional attribute (fourth argument) is the sample rate; it defines how many samples must be played per second in order to reconstruct the original sound.

P.S. &samples[0] – указатель на начало данных (лежат последовательно).  
Запись равносильна samples.data()

$$\text{samples}[i] = A \sin\left(2\pi f \frac{i}{\text{channels} * \text{sampleRate}}\right), A \approx 5 \times 10^4, \text{sampleRate} = 44 \times 10^3, f = 440 \text{Гц}$$

# sf::Sound

- Now that the audio data is loaded, we can play it with a [sf::Sound](#) instance.

```
sf::SoundBuffer buffer;  
// Load something into the sound buffer...  
  
sf::Sound sound;  
sound.setBuffer(buffer); sound.setLoop(true);  
sound.play();  
  
while (true) { ... }
```

# Упражнение 2 («синтезатор»)

- Для каждой клавиши своя частота «гудения» (ля == 440 Гц)

Полная стандартная шкала частот музыкальных тонов

Нота	Частота, Гц								
	Суб-контр-октава	Контр-октава	Большая октава	Малая октава	Первая октава	Вторая октава	Третья октава	Четвертая октава	Пятая октава
До (В)		32,70	65,41	130,82	261,63	523,25	1046,5	2093,0	4186,0
До-диез (С <sup>♯</sup> )		34,65	69,30	138,59	277,18	554,36	1108,7	2217,4	4434,8
Ре (D)		36,95	73,91	147,83	293,66	587,32	1174,6	2349,2	4698,4
Ре-диез (D <sup>♯</sup> )		38,88	77,78	155,56	311,13	622,26	1244,5	2489,0	4978,0
Ми (E)	20,61	41,21	82,41	164,81	329,63	659,26	1318,5	2637,0	5274,0
Фа (F)	21,82	43,65	87,31	174,62	349,23	698,46	1396,9	2793,8	
Фа-диез (F <sup>♯</sup> )	23,12	46,25	92,50	185,00	369,99	739,98	1480,0	2960,0	
Соль (G)	24,50	49,00	98,00	196,00	392,00	784,00	1568,0	3136,0	
Соль-диез (G <sup>♯</sup> )	25,95	51,90	103,80	207,60	415,30	830,60	1661,2	3332,4	
Ля (A)	27,50	55,00	110,00	220,00	440,00	880,00	1720,0	3440,0	
Ля-диез (В)	29,13	58,26	116,54	233,08	466,16	932,32	1864,6	3729,2	
Си (H)	30,87	61,74	123,48	246,96	493,88	987,75	1975,5	3951,0	

# Упражнение 2 («синтезатор»)

- Длительность гудения клавиши фиксирована (например, 0.5 сек)
- Громкость гудения плавно уменьшается (например, с помощью метода ***setVolume***)
- Звуки от разных клавиш могут звучать одновременно (mixing)

P.S. Можно программно сгенерировать затухающее **Ля**, а остальные частоты создать на её основе с помощью ***setPitch*** (Changing the pitch has a side effect: it impacts the playing speed)



# Запись с микрофона

- <http://www.sfml-dev.org/tutorials/2.3/audio-recording.php>

# Метод Stop у MediaPlayer

Остановить работу потока, где проигрывается музыка (метод Stop), можно, например, так:

<http://pastebin.com/3HhWMSTH>