Объектно-ориентированное программирование

Лекция 4. Полиморфизм и пр.

Полиморфизм

```
class Base {
private:
    int x = 0;
public:
    virtual void print_x(){};
};

class A : public Base{
public:
    void print_x() override {
        std::cout << "A.x = " << x << "\n";
    };
private:
    int x = 1;
};
```

```
std::vector <Base*> base_vec;
A a;
base_vec.push_back(&a);
```

Полиморфизм – это свойство, которое позволяет одно и тоже имя использовать для решения нескольких технически разных задач.

Дружественные функции

```
class A {
public:
    friend void f(A&);
    void print_x();
private:
    int x;
};

void f(A& a) {
    a.x = 10;
}

int main()
{
    A a;
    f(a);
}
```

Дружественные функции - это функции, которые не являются членами класса, однако имеют доступ к его закрытым членам - переменным и функциям, которые имеют спецификатор private.

В качестве дружественной функции могут выступать методы другого класса

Возможное применение – одна функция работающая со многими классами.

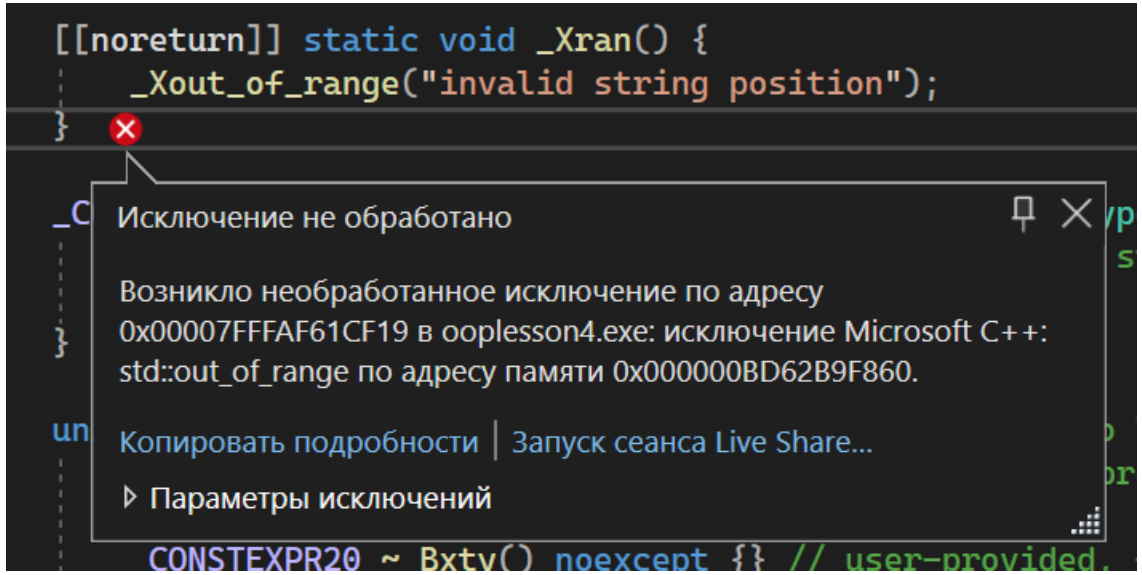
Дружественные классы

```
class A {
    friend class B;
private:
    int x;
};

class B {

public:
    void print_x(A& a) {
        std::cout << "X = " << a.x << "\n";
    }
    void print_y() {
        std::cout << "Y = " << y << "\n";
    }
private:
    int y;
};
```

При объявлении класса можно объявить сразу все функции-члены другого класса дружественными одним объявлением. Таким образом создается **дружественный класс**.

Обработка исключений

```cpp
#include <iostream>
#include <string>

int main() {
    std::string s = "Hello world!";
    std::cout << s.at(25) << std::endl;
    return 0;
}
```



try-catch - это механизм обработки исключений в C++, который позволяет обрабатывать ошибки и исключения, возникающие во время выполнения программы.

Обработка исключений

```
#include <iostream>
#include <string>

int main() {
    try {
        std::string s = "Hello world!";
        std::cout << s.at(25) << std::endl;
    }
    catch (std::out_of_range e) {
        std::cerr << "Caught an out_of_range exception: " << e.what() <<
std::endl;
    }
    return 0;
}
```

Обертка **try-catch** предотвращает падение программы

Создание исключений

throw в C++ используется для генерации исключений.

```
#include <iostream>
#include <string>

int main() {
    try {
        std::string s = "Hello world!";
        std::cout << s.at(2) << std::endl;
        throw std::runtime_error("Hello! Im a runtime error");
        throw std::logic_error("Hello! Im a logic error");
    }
    catch (std::out_of_range e) {
        std::cerr << "Caught an out_of_range exception: " << e.what() << std::endl;
    }
    catch (std::logic_error e) {
        std::cerr << "Caught an throw logic error: " << e.what() << std::endl;
    }
    catch (std::runtime_error e) {
        std::cerr << "Caught an throw runtime error: " << e.what() << std::endl;
    }
    return 0;
}
```