

Министерство образования Республики Беларусь
Учреждение образования
“Брестский Государственный технический университет”
Кафедра ИИТ

Лабораторная работа №6

По дисциплине “Проектирование программного обеспечения интеллектуальных систем”
Тема: “Пространства имен. Исключения”

Выполнил:
Студент 2 курса
Группы ИИ-21
Кирилович А. А.
Проверил:
Монтик Н. С.

Цель работы: знакомство с использованием пространств имен. Знакомство с исключениями.

Ход работы: Разработать три пользовательских класса в соответствии с целью.

```
#include <iostream>

namespace Device {
    class USBDevice {
    private:
        int m_id;
        int m_type;
    public:
        USBDevice() {
            m_id = 0;
            m_type = 0;
        }
        USBDevice(int id, int type)
            : m_id(id), m_type(type) {}
        USBDevice(const USBDevice& other)
            : m_id(other.m_id), m_type(other.m_type) {}
        ~USBDevice() {}
        void SetId(int id) { m_id = id; }
        void SetType(int type) { m_type = type; }
        int GetId() { return m_id; }
        int GetType() { return m_type; }
        void Print() {
            std::cout << "USBDevice: id = " << m_id << ", type = " << m_type << std::endl;
        }
    };
}

namespace Hub {
    class USBHub {
    private:
        int m_id;
        int m_type;
        int m_ports;
    public:
        USBHub() {
            m_id = 0;
            m_type = 0;
            m_ports = 0;
        }
        USBHub(int id, int type, int ports)
            : m_id(id), m_type(type), m_ports(ports) {}
        USBHub(const USBHub& other)
            : m_id(other.m_id), m_type(other.m_type), m_ports(other.m_ports) {}
        ~USBHub() {}
        void SetId(int id) { m_id = id; }
        void SetType(int type) { m_type = type; }
        void SetPorts(int ports) { m_ports = ports; }
        int GetId() { return m_id; }
        int GetType() { return m_type; }
        int GetPorts() { return m_ports; }
        void Print() {
            std::cout << "USBHub: id = " << m_id << ", type = " << m_type << ", ports = " << m_ports <<
std::endl;
        }
    };
}

namespace DeviceAndHub
{
    class USBDeviceHub : public Device::USBDevice, public Hub::USBHub {
    public:
        USBDeviceHub() {
            USBHub::SetId(0);
            USBHub::SetType(0);
            USBHub::SetPorts(0);
        }
        USBDeviceHub(int id, int type, int ports)
            : USBDevice(id, type), USBHub(id, type, ports) {}
        USBDeviceHub(const USBDeviceHub& other)
            : USBDevice(other), USBHub(other) {}
        ~USBDeviceHub() {}
        void SetId(int id) { USBDevice::SetId(id); USBHub::SetId(id); }
    };
}
```

```

void SetType(int type) { USBDevice::SetType(type); USBHub::SetType(type); }
void SetPorts(int ports) { USBHub::SetPorts(ports); }
int GetId() { return USBDevice::GetId(); }
int GetType() { return USBDevice::GetType(); }
int GetPorts() { return USBHub::GetPorts(); }
void Print() {
    std::cout << "USBDeviceHub: id = " << USBDevice::GetId() << ", type = " << USBDevice::GetType() <<
", ports = " << USBHub::GetPorts() << std::endl;
}
};

int main() {
    Device::USBDevice device(1, 2);
    Hub::USBHub hub(3, 4, 5);
    DeviceAndHub::USBDeviceHub devicehub(6, 7, 8);
    int arr[3] = { 1, 2, 3 };
    try {
        for (int i = 0; i < 4; i++) {
            std::cout << arr[i] << std::endl;
        }
    }
    catch(std::bad_alloc e) {
        std::cerr << e.what() << '\n';
    }

    device.Print();
    hub.Print();
    devicehub.Print();
}

```

```

1
2
3
-1548166144
USBDevice: id = 1, type = 2
USBHub: id = 3, type = 4, ports = 5
USBDeviceHub: id = 6, type = 7, ports = 8

```