Исключения. Методы обработки исключений

Исключение

- Исключение – это непредвиденное событие, делающее невозможным дальнейшее выполнение программы по базовому алгоритму.

Виды исключений

a) Синхронные исключения – такие исключения могут возникнуть только в определенных точках программы.

a) Асинхронные исключения – могут возникать в любой момент и независимо от того, что выполняет программа.

Обработка исключений

1. Обнаружение исключения
2. Передача управления обработчику исключений
3. Выполнение обработчика

- Обработчик исключений – это блок кода, которому передается управление при возникновении исключения.

Виды обработки исключений

а) Обработка с возвратом – обработчик ликвидирует возникшее исключение и передает управление в ту точку программы, где возникло исключение

а) Обработка без возврата – после выполнения кода обработчика, управление возвращается в заранее заданное место программы.

Целесообразность обработки исключений

Исключения стоит использовать в следующих случаях:

a) Заранее известно, что невозможно обработать исключение таким образом, чтобы программа смогла продолжить выполнение алгоритма.

b) Необходимо уведомить пользователя о невозможности корректно создать объект без особого формата ввода данных.

Исключения не стоит использовать по следующим причинам:

a) Исключения не предназначены для организации нормальных путей выполнения программы.

b) Данный механизм снижает быстродействие программы.

c) Пользователю необходимо знать про исключения, использующиеся в программе

Механизм обработки исключений в C++

```
try // контролируемый блок
{
//исключение
throw()//генерация объекта или передача значения


}
catch()
{//обработка исключения или вывод сообщения об ошибке
}
```

Контролируемый блок try

Блок *try* используется для проверки возникновения исключения, соответственно в блок *try* помещается та часть кода, в которой возможно возникновение исключения.

Свойства блока *try*

a) Связан с одним или несколькими операторами *catch*

b) Все переменные объявленные внутри *try* являются локальными

Оператор throw

Оператор *throw* - передает заданный объект в один из операторов *catch*.

Форма записи:

throw(объект); или *throw* объект; , где объектом может быть

- Константа
- Переменная
- Выражение

Тип объекта может быть как встроенным, так и определяемым пользователем.

Оператор *catch*

Оператор *catch* – принимает объект, переданный оператором *throw,* и после передачи объекта выполняется код написанный в данном операторе.

Форма записи:

catch(<тип> <имя объекта>) *catch (int i)*

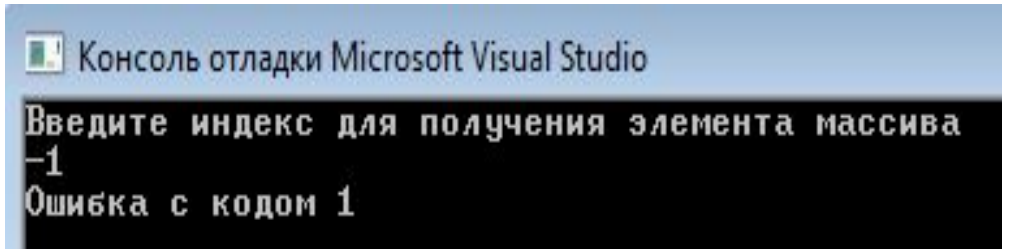
catch(<тип объекта>) *catch (int)*

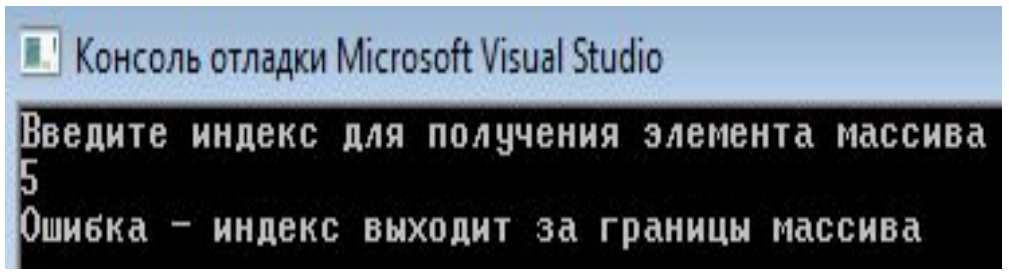
catch(…)

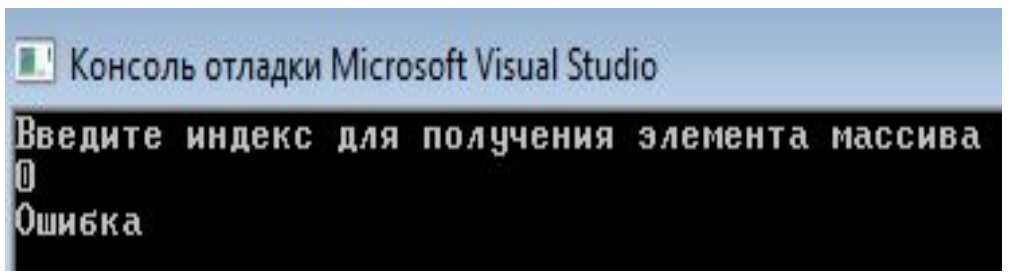
- Для всех возможных типов объектов, которые может передать *throw*, должны существовать операторы *catch*, иначе компилятор выдаст ошибку “Необработанное исключение”.

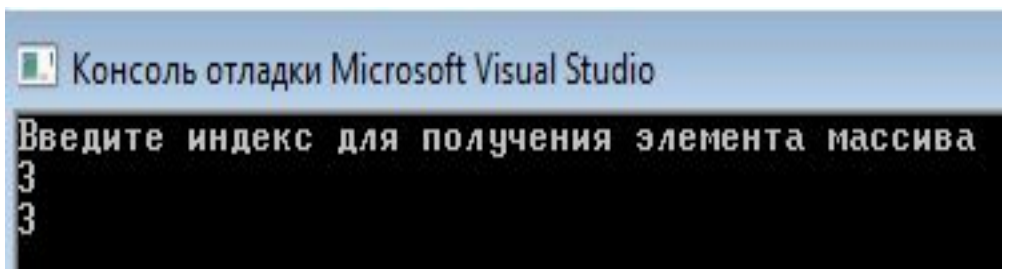
- *catch(*<тип> <имя объекта>) – при такой форме записи оператор принимает сам объект, и переданный объект может использоваться в самом блоке *catch*. Принять можно по ссылке, копию объекта или указатель на объект.
- *catch(*<тип объекта>) - при такой форме записи оператор не принимает сам объект, соответственно переданный объект не получится использовать в блоке *catch*.
- *catch (...)* - при такой форме записи оператор принимает любые объекты, передаваемые оператором *throw*.

```
try
{
    cout << "Введите индекс для получения элемента массива" << endl;
    cin >> index;
    if (index < 0) throw 1;
        if (index > 4) throw '2';
            if (index == 0) throw "3";
            cout << a[index] << endl;
}
catch (int& i)
{
    cout << " Ошибка с кодом " << i << endl;
}
catch (char)
{
    cout << "Ошибка - индекс выходит за границы массива"<<endl;
}
catch (...)
{
    cout << " Ошибка" << endl;
}
```









- Порядок *catch* операторов имеет значение, при данной расстановке операторов, появится ошибка “маскирование обработ

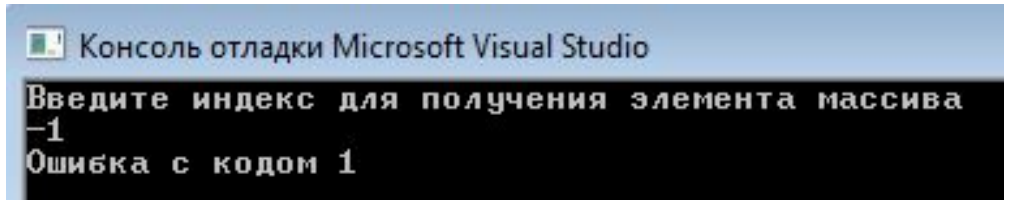
```
try
{
    cout << "Введите индекс для получения элемента массива" << endl;
    cin >> index;
    if (index < 0) throw 1;
        if (index > 4) throw '2';
            if (index == 0) throw "3";
            cout << a[index] << endl;
}
catch (int& i)
{
    cout << "Ошибка с кодом " << i << endl;
}
catch (...)
{
    cout << "Ошибка" << endl;
}
catch (char)
{
    cout << "Ошибка - индекс выходит за границы массива"<<endl;
}
```

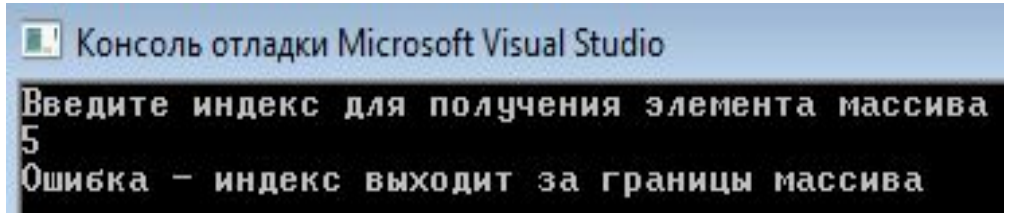
Обработка исключений с помощью функций

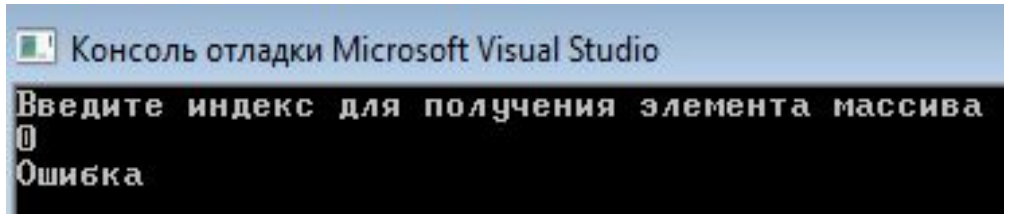
```
try
{
    cout << "Введите индекс для получения элемента массива" << endl;
    cin >> index;
    except(index);
    cout << a[index] << endl;
}
catch (int& i)
{
    cout << "Ошибка с кодом " << i << endl;
}
catch (char)
{
    cout << "Ошибка - индекс выходит за границы массива" << endl;
}
catch (...)
{
    cout << "Ошибка" << endl;
}
```

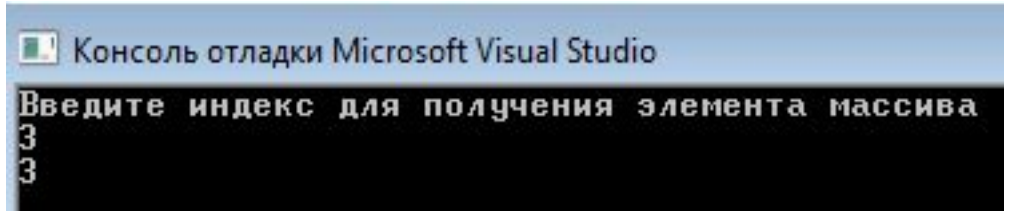
```
void except(int& index)
{
    if (index < 0) throw 1;
        if (index > 4) throw '2';
            if (index == 0) throw "3";
                if (index == 1)throw 3.1;

}
```

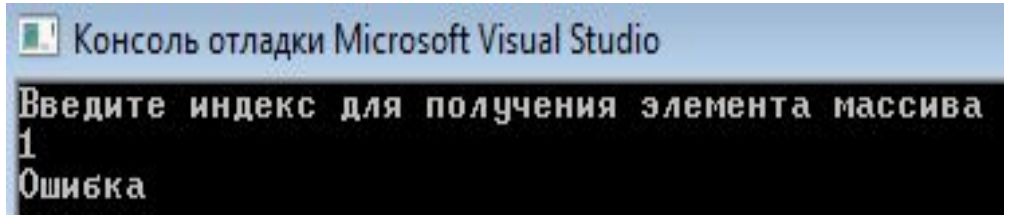
```
void except(int& index) throw();
void except(int& index) throw(int,string,double,char);
void except(int& index);
```


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
-1
Ошибка с кодом 1


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
5
Ошибка - индекс выходит за границы массива


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
0
Ошибка


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
3
3


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
1
Ошибка

Обработка исключений с помощью функций

- Если не найдено соответствующего оператора для генерируемого функцией исключения, то компилятор вызывает функцию unexpected(), которая вызывает функцию terminate(), завершающую программу. Обе функции можно подменить своими реализациями, для этого

1. Подключить #include<expection>
2. Определить собственную функцию с прототипом void F()
3. Для подмены terminate(), написать set_terminate(F), такая функция не должна возвращать исключения или управление оператором return. Она может только завершить программу функцией exit(int) или abort()

Для подмены unexpected(), set_unexpected(F).

Реализация собственной иерархии исключений

1. Определяем базовый класс:

```
class error
{
protected:
    string message;
    int number;
public:
    virtual void what() = 0;
};
```

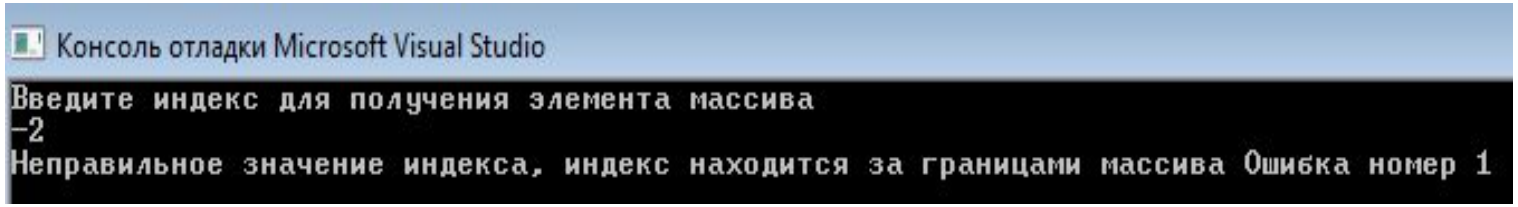
2. Определяем производные классы, которые будут содержать в себе описание конкретного исключения.

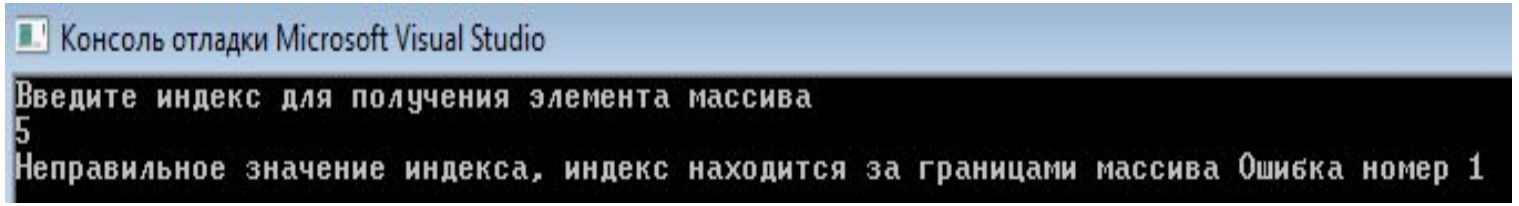
```
class error_size: public error
{
public:
    error_size()
    {
        number = 1;
        message = "Неправильное значение индекса, индекс находится за границами массива Ошибка номер ";
    };
    error_size(const error_size& a)
    {
        number = a.number;
        message = a.message;
    };
    void what()
    {
        cout << message <<number<< endl;
    }
};
```

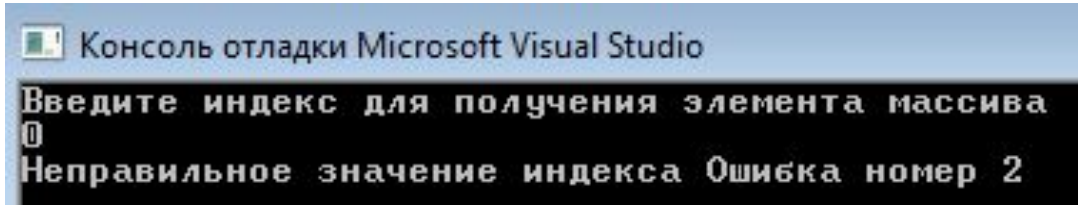
```
class error_index : public error
{
public:
    error_index()
    {
        number = 2;
        message = "Неправильное значение индекса Ошибка номер ";
    };
    error_index(const error_index& a)
    {
        number = a.number;
        message = a.message;
    };
    void what()
    {
        cout << message << number << endl;
    }
};
```

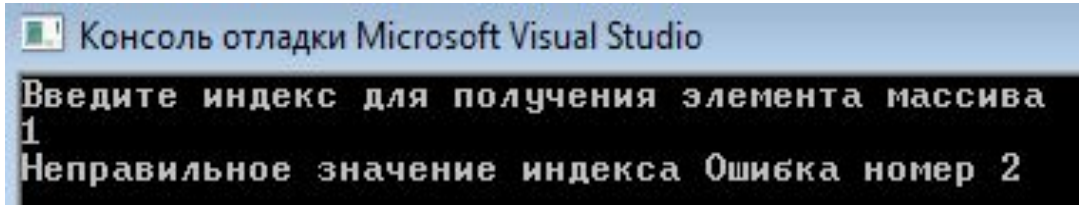
```
try
{
    cout << "Введите индекс для получения элемента массива" << endl;
    cin >> index;
    except(index);
    cout << a[index] << endl;
}
catch (error& a)
{
    a.what();
}
catch (...)
{
    cout << "Сгенерировано непредвиденное исключение" << endl;
}
```

```
void except(int& index)
{
    if (index < 0) throw error_size();
        if (index > 4) throw error_size();
            if (index == 0) throw error_index();
                if (index == 1)throw error_index();

}
```


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
-2
Неправильное значение индекса, индекс находится за границами массива Ошибка номер 1


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
5
Неправильное значение индекса, индекс находится за границами массива Ошибка номер 1


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
0
Неправильное значение индекса Ошибка номер 2


Консоль отладки Microsoft Visual Studio
Введите индекс для получения элемента массива
1
Неправильное значение индекса Ошибка номер 2

Разбор 15 варианта лабораторной работы

1. Определяем базовый класс:

```
class error
{
protected:
    string message;
public:
    void what() { cout << message; }
};
```

2. Определяем производные классы, которые будут содержать в себе описание конкретного исключения.

```
class error_index_min : public error
{
public:
    error_index_min()
    {
        message = "Index fewer than zero";
    };
};
```

```
class error_index_max : public error
{
public:
    error_index_max()
    {
        message = "Index larger than max index";
    };
};
```

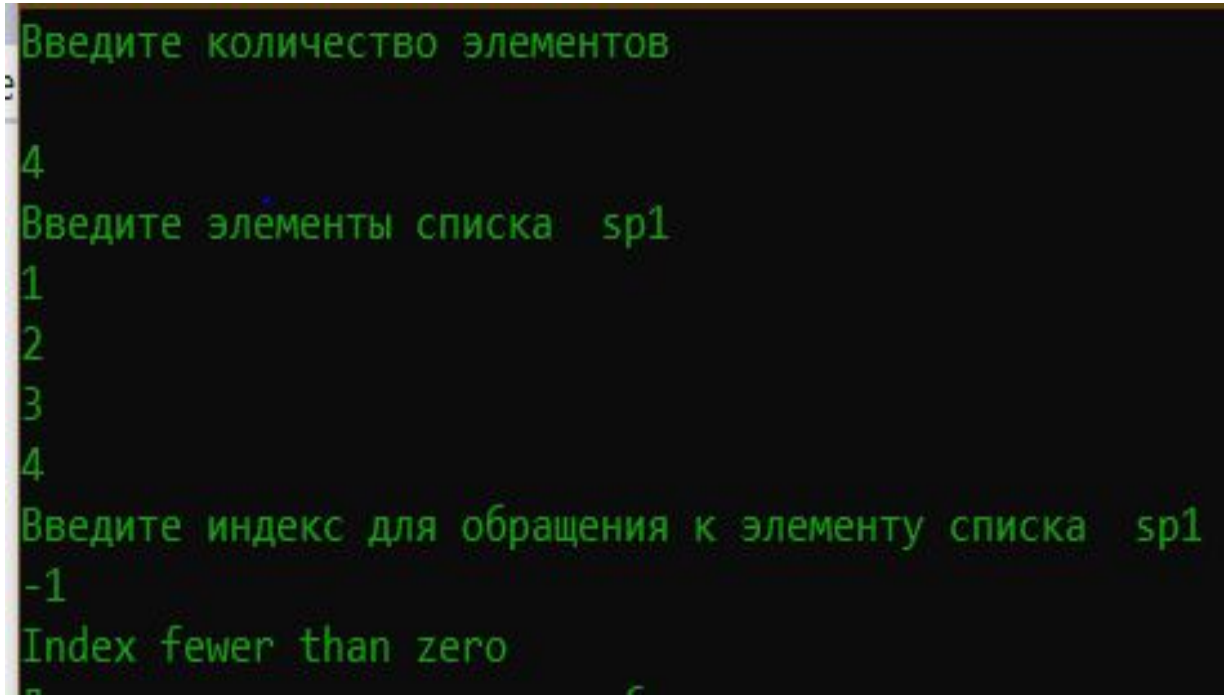
3. Добавляем *throw* в соотвествующие методы.

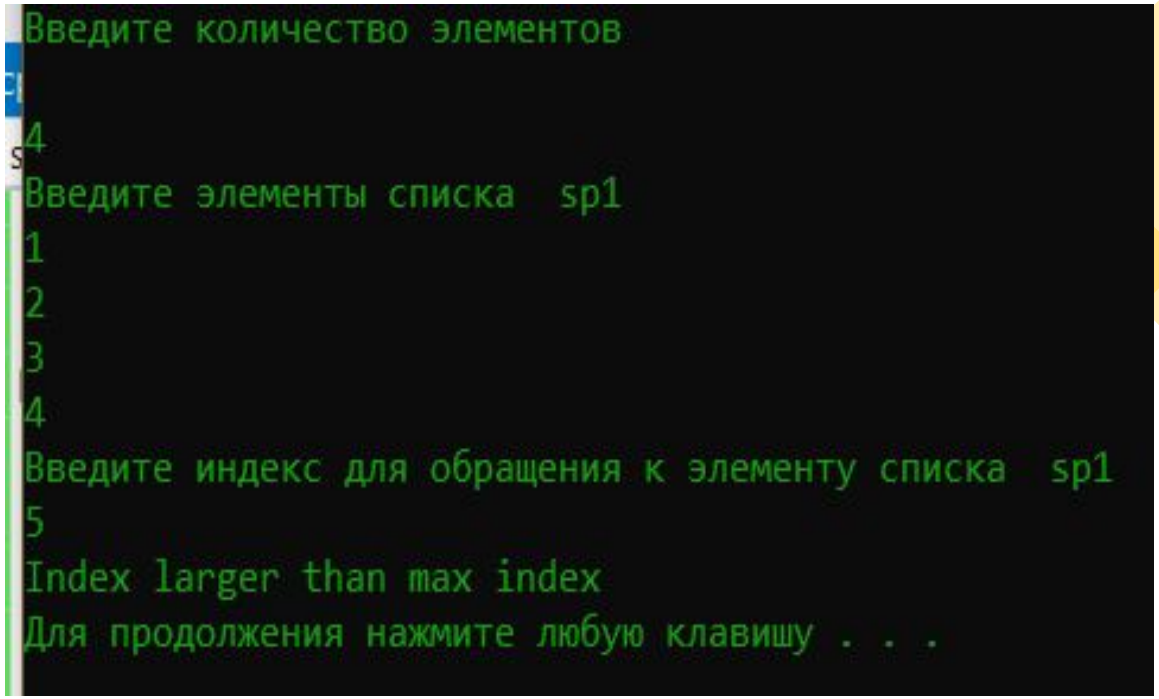
```
int spisok::operator[](int n)
{
    if (n < 0)throw error_index_min();
    if (n >= size)throw error_index_max();
    elem* c = head;
    for (int i = 0; i < n; ++i)
        c = c->next;
    return (c->data);
}
```

4. Помещаем код в блок try.

```
try
{
    cout << "Введите количество элементов" << endl << endl;
    cin >> count;
    if (count < 1) throw "Count must be larger than zero";
    cout << "Введите элементы списка  sp1" << endl;
    for (int i = 0; i < count; ++i)
    {
        cin >> x;
        sp1.push_back(x);
    }
    iter.begin(sp1);
    cout << "Введите индекс для обращения к элементу списка  sp1" << endl;
    cin >> index;
    cout << sp1[index] << endl;
    cout << "Введите индекс для перехода к элементу списка  sp1" << endl;
    cin >> index;
    iter + index;
    cout << iter.get() << endl;
}
catch (error& a)
{
    a.what();
}
```

5.Результат.


Введите количество элементов
4
Введите элементы списка sp1
1
2
3
4
Введите индекс для обращения к элементу списка sp1
-1
Index fewer than zero


Введите количество элементов
4
Введите элементы списка sp1
1
2
3
4
Введите индекс для обращения к элементу списка sp1
5
Index larger than max index
Для продолжения нажмите любую клавишу . . .