Dynamic memory
Exception handling

CSE2013-17F

Dynamic memory allocation

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

int main(void) {
    int size;
    cout << "Size of array? ";
    cin >> size;

    int* arr = (int*)malloc(sizeof(int)*size);

    for (int i = 0; i < size; i++)
        arr[i] = i + 10;
    for (int j = 0; j < size; j++)
        cout << "arr[" << j << "]=" << arr[j] << endl;
    free(arr);

    return 0;
}
```

- ***C***: malloc, free

```
Size of array? 5
arr[0]=10
arr[1]=11
arr[2]=12
arr[3]=13
arr[4]=14
```

Dynamic memory allocation

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

int main(void) {
    int size;
    cout << "Size of array? ";
    cin >> size;

    int* arr = new int[size];

    for (int i = 0; i < size; i++)
        arr[i] = i + 10;
    for (int j = 0; j < size; j++)
        cout << "arr[" << j << "]=" << arr[j] << endl;
    delete[] arr;

    return 0;
}
```

- ***C++***: new, delete

```
Size of array? 5
arr[0]=10
arr[1]=11
arr[2]=12
arr[3]=13
arr[4]=14
```

Dynamic memory allocation

```
// memory allocation for an int data
int* val = new int;

// memory allocation for the int array of length ‘size’
int* arr = new int[size];

// deallocates ‘val’
delete val;

// deallocates ‘arr’
delete[] arr;
```

Dynamic memory allocation

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

int main(void) {
    int size;
    cout << "Size of array? ";
    cin >> size;

    int* arr = new int[size];
    if (arr == NULL) {
        cout << " Failed to allocate memory" << endl;
        return -1;    // Program exit
    }
    for (int i = 0; i < size; i++)
        arr[i] = i + 10;
    for (int j = 0; j < size; j++)
        cout << "arr[" << j << "]=" << arr[j] << endl;
    delete[] arr;

    return 0;
}
```

- If there is not enough memory space to allocate, the *new* keyword returns *NULL*.

Exception handling

- Traditional style: if, else ...
- C++ exception handling
  - **try**: A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.
  - **catch**: A program catches an exception with an exception handler at the place in a program where you want to handle the problem.
  - **throw**: A program throws an exception when a problem shows up.

Exception handling

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

int main(void) {
    int a, b;

    cout << "Input two numbers: ";
    cin >> a >> b;

    try {
        if (b == 0)
            throw b;
        cout << "a / b = " << a / b << endl;
        cout << "a % b = " << a%b << endl;
    }
    catch (int exception) {
        cout << "Input " << exception << endl;
        cout << "Input error!" << endl;
    }
    return 0;
}
```

```
Input two numbers: 5 2
a / b = 2
a % b = 1
```

```
Input two numbers: 5 0
Input 0
Input error!
```

Exception handling

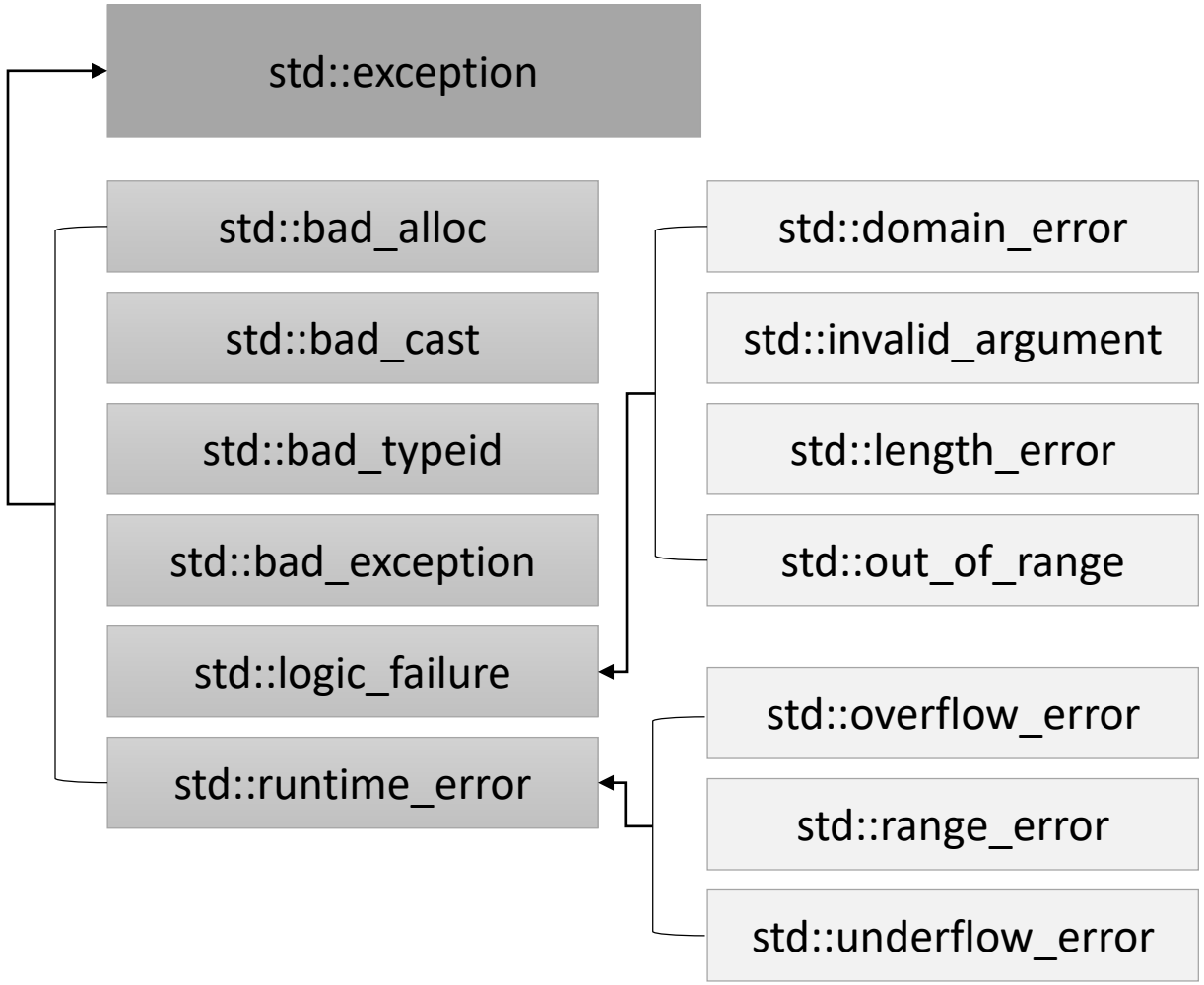
```
#include <iostream>
using namespace std;

int main(void) {
    try {
        int i = 0;
        while (1) {
            cout << i++ << " is allocated" << endl;
            double(*arr)[1000] = new double[1000][1000];
        }
    }
    catch (exception ex) {
        cout << ex.what();
        cout << endl << "END" << endl;
    }
    return 0;
}
```

```
...
254 is allocated
255 is allocated
256 is allocated
257 is allocated
258 is allocated
259 is allocated
bad allocation
END
```

Exception handling

- std::exception



Exception handling

- Exception class

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

class MyEx
{
    char str[10];
public:
    MyEx(char* input) {
        strcpy_s(str, input);
    }
    void What() {
        cout << str << endl;
        cout << "My Exception" << endl;
    }
};

int main(void) {
    try {
        throw MyEx("Hello");
    }
    catch (MyEx& ex) {
        ex.What();
    }
    return 0;
}
```