Assignment No. 09

- **TITLE :** Implement a program to handle arithmetic exceptions and array index out of bounds.
- **CODE:**

1. Using Built-in Exceptions

```
#include <iostream>
#include <exception>
#include <stdexcept>
#include <vector>
using namespace std;

void divByZero(int a, int b){
    if (b == 0){
        throw runtime_error("Division by zero");
    }
    int res = a / b;
    cout << "The quotient is " << res << endl;
}

void accessArrayAt(int idx){
    vector<int> arr = {10, 20, 30};
    cout << "Element at index " << idx << " is " << arr.at(idx) << endl;
}

int main(){
    try {
        divByZero(10, 0);
    }
    catch (const runtime_error &e) {
        cout << "Division error: " << e.what() << endl;
    }

    try {
        accessArrayAt(5);
    }
    catch (const out_of_range &e) {
        cout << "Array access error: " << e.what() << endl;
    }
    return 0;
}
```

Output :

```
Division error: Division by zero
Array access error: vector::_M_range_check: __n (which is 5) >= this->size() (which is 3)
```

---

2. Using STDEXECPT Class

```
#include <iostream>
#include <exception>
#include <stdexcept>
```

```
#include <vector>
#include <string>
using namespace std;

class DivisionByZero : public runtime_error {
public:
    DivisionByZero(const string& msg = "Division by 0 is not allowed")
        : runtime_error(msg) {}
};

class ArrayOutOfBounds : public out_of_range {
public:
    ArrayOutOfBounds(const string& msg = "Accessed element of array which is out
of bounds")
        : out_of_range(msg) {}
};

int divide(int a, int b){
    if (b == 0) {
        throw DivisionByZero();
    }
    return a / b;
}

void arrOutOfBound(vector<int> arr){
    try {
        arr.at(10);
    }
    catch (const out_of_range &e) {
        throw ArrayOutOfBounds(e.what());
    }
}

int main(){
    int ch;
    vector<int> arr = {1,2,3,4,5};
    int a,b,res;
    while(true){
        cout<<"Choose an option!"<<endl;
        cout<<"1.Division\n2.Array\n3.Exit"<<endl;
        cin>>ch;
        switch (ch)
        {
        case 1:
            cout<<"Enter numerator and denominator: "<<endl;
            cin>>a>>b;
            try {
                res = divide(a,b);
                cout<<"Result: "<<res<<endl;
            }
            catch (const DivisionByZero &d) {
                cout<<"Division Error: "<<d.what()<<endl;
            }
            break;
        case 2:
            try {
                arrOutOfBound(arr);
                cout<<"Array access succeeded."<<endl;
            }
            catch (const ArrayOutOfBounds &r) {
                cout<<"Array Error: "<<r.what()<<endl;
            }
            break;
        case 3:
            cout<<"Exitting...."<<endl;
            return 0;
        default:
            cout<<"Invalid Choice!"<<endl;
            break;
        }
    }
    return 0;
}
```

Output :

```
Choose an option!
1.Division
2.Array
3.Exit
1
Enter numerator and denominator:
10 0
Division Error: Division by 0 is not allowed
Choose an option!
1.Division
2.Array
3.Exit
2
Array Error: vector::_M_range_check: __n (which is 10) >= this->size() (which is 5)
Choose an option!
1.Division
2.Array
3.Exit
3
Exitting....
```

---

3. Using User-Defined Exceptions

```
#include <iostream>
#include <vector>
#include <string>
#include <exception>
using namespace std;

class DivisionByZeroException : public exception
{
    string msg;
```

```
public:
    DivisionByZeroException(const string &m = "Division by zero") : msg(m) {}
    const char *what() const noexcept override {
        return msg.c_str();
    }
};

class ArrayOutOfBoundsException : public exception
{
    string msg;

public:
    ArrayOutOfBoundsException(const string &m = "Array index out of bounds") :
msg(m) {}
    const char *what() const noexcept override {
         return msg.c_str();
        }
};

int divide(int a, int b)
{
    if (b == 0)
        throw DivisionByZeroException();
    return a / b;
}

int accessAt(const vector<int> &arr, int idx)
{
    if (idx < 0 || idx >= static_cast<int>(arr.size()))
        throw ArrayOutOfBoundsException();
    return arr[idx];
}

int main()
{
    vector<int> arr = {1, 2, 3, 4, 5};
    int choice = 0;
    while (true)
    {
        cout << "\nChoose an option:\n1. Division\n2. Array access\n3.
Exit\nChoice: ";
        if (!(cin >> choice))
        {
            cout << "Invalid input. Exiting.\n";
            return 1;
        }
        if (choice == 1)
        {
            int a, b;
            cout << "Enter numerator and denominator: ";
            cin >> a >> b;
            try
            {
                int res = divide(a, b);
                cout << "Result: " << res << endl;
            }
            catch (const DivisionByZeroException &e)
            {
                cout << "Division Error: " << e.what() << endl;
            }
        }
        else if (choice == 2)
        {
            int idx;
            cout << "Enter index to access (0 to " << arr.size() - 1 << "): ";
            cin >> idx;
            try
            {
                int val = accessAt(arr, idx);
                cout << "Element at index " << idx << " is " << val << endl;
            }
            catch (const ArrayOutOfBoundsException &e)
            {
                cout << "Array Error: " << e.what() << endl;
            }
        }
        else if (choice == 3)
        {
            cout << "Exiting...\n";
            break;
        }
        else
        {
            cout << "Invalid choice.\n";
        }
    }
    return 0;
}
```

Output :

```
Choose an option:
1. Division
2. Array access
3. Exit
Choice: 1
Enter numerator and denominator: 10 0
Division Error: Division by zero

Choose an option:
1. Division
2. Array access
3. Exit
Choice: 2
Enter index to access (0 to 4): 7
Array Error: Array index out of bounds
```

```
Choose an option:
1. Division
2. Array access
3. Exit
Choice: 3
Exiting...
```

- **Conclusion –** In this assignment, we've implemented program on implementation of exception handling in C++.