Exceptions in other word is error handler. Mostly you want to check the input format from user is correct or not.

For example:

If you prompt user to input an integer number, but user input a character or string. It will sometimes make your program crush.

So the concept is that:

Try –> if statement -> throw the argument to the catch function(usually a message) -> catch the argument and do the corresponding jobs.

Notice: you can do the same error input check by just an “if” statement.

Let’s see the example:

Input Error check by exception ( **try, throw, catch** )

#include <iostream>

#include <string>

using namespace std;

int main()

{

int age = 0;

string err\_message = "positive integer only plz!";

string trash;

do{

try{

cout << "Enter your age: ";

cin >> age;

if (cin.fail() || age <= 0) throw err\_message;

cout << "You are " << age << " year old!" << endl;

}

catch (string s)

{

cout << s << endl;

cin.clear();

getline(cin, trash);

}

} while (age <= 0);

system("pause");

return 0;

}

In this case, I firstly made an error message (string type). Then I prompt user to input their age (expect an integer). However, you never know what input user will give you. Someone maybe give you a string, a character, or a float number. So I have the if statement:

if (cin.fail() || age <= 0) throw err\_message;

This if statement will determine the input is valid or not.

cin.fail() happened when the input from user is not an integer. If it is not an integer, but I’m trying to cin into an integer type variable (age). Now the system will set a fail flag in the file stream to mention your cin instruction doesn’t work. Another condition age <= 0 is to check the input age is a positive integer(age should be positive). If the situation happen, I throw the error message to the catch function.

Since I’m sending the string type parameter to the catch, the catch function need to be defined to have a string parameter:

catch (string s)

{

cout << s << endl;

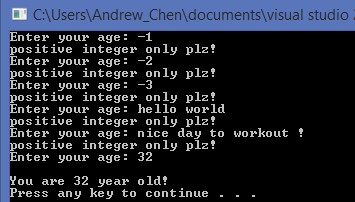
cin.clear();

getline(cin, trash);

}

In this catch function, I firstly output the error message from the parameter I through above. Then reset the fail bit in the string and clean the cin stream(the previous invalid input from user is still in the buffer so you need to clean it. The way I do is get the whole line into a string variable.)

Output:



As I mention before, you can also do this by only if statement:

So here is the example:

#include <iostream>

#include <string>

using namespace std;

int main()

{

int age = 0;

string err\_message = "positive integer only plz!";

string trash;

do{

cout << "Enter your age: ";

cin >> age;

if (cin.fail() || age <= 0)

{

cin.clear();

getline(cin, trash);

cout << "your age must be an integer which greater than 0!" << endl;

}

else cout << "you are " << age << " years old" << endl;

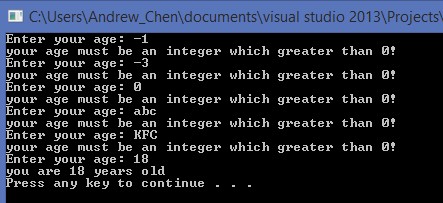
} while (age <= 0);

system("pause");

return 0;

}

Output:



Another thing for exception by try, throw, catch. Sometime you may use nesting exception. You have a try inside another try. And remember for each “**throw**”, you must need to have a “**catch**” with the correct type of parameter to work with the element you throw.

Example:

int main()

{

try{

int numerator, denominator;

cout << "Enter a integer number of numerator: ";

cin >> numerator;

cout << "Enter a integer number of denominator: ";

cin >> denominator;

if (denominator == 0) throw "denominator can not be 0!";

cout << numerator << " divided by " << denominator << " is "

<< numerator / double(denominator) << endl;

try

{

if (numerator + denominator == 0) throw 0;

cout << "The sum is: " << numerator + denominator << endl;

}

catch (int n)

{

cout << "the sum is 0!" << endl;

}

}

catch (char \*c)

{

cout << c << endl;

}

system("pause");

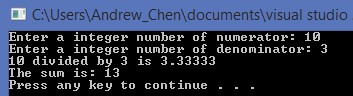
return 0;

}

Let’s see different situation:

1. If the two input are both valid (integer and their sum != 0) The throw and catch will not be use in the process. So the output will show you the fraction and sum.

Output:



1. If the denominator is 0, the first if statement will check for it and do the things below:
2. Throw a cstring out.
3. Break the first try {}.

Then the program will jump to the remain of code to find a catch function which match the cstring parameter to work with the throw.

Your code will run like this:

int main()

{

try{

int numerator, denominator;

cout << "Enter a integer number of numerator: ";

cin >> numerator;

cout << "Enter a integer number of denominator: ";

cin >> denominator;

if (denominator == 0) **throw** **"denominator can not be 0!"**;

cout << numerator << " divided by " << denominator << " is "

<< numerator / double(denominator) << endl;

try

{

if (numerator + denominator == 0) throw 0;

cout << "The sum is: " << numerator + denominator << endl;

}

catch (int n)

{

cout << "the sum is 0!" << endl;

}

}

**catch (char \*c)**

**{**

**cout << c << endl;**

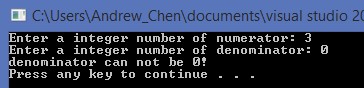
**}**

system("pause");

return 0;

}

Output will just print the cstring you throw:



1. If the first try is good ( denominator != 0 ), but the second try does not pass the condition

if (numerator + denominator == 0) throw 0;

Then stop the second try and throw an integer 0. Keep running the code below the second try to find a catch function match integer parameter:

int main()

{

try{

int numerator, denominator;

cout << "Enter a integer number of numerator: ";

cin >> numerator;

cout << "Enter a integer number of denominator: ";

cin >> denominator;

if (denominator == 0) throw "denominator can not be 0!";

cout << numerator << " divided by " << denominator << " is "

<< numerator / double(denominator) << endl;

try

{

if (numerator + denominator == 0) **throw 0**;

cout << "The sum is: " << numerator + denominator << endl;

}

**catch (int n)**

**{**

**cout << "the sum is 0!" << endl;**

**}**

}

catch (char \*c)

{

cout << c << endl;

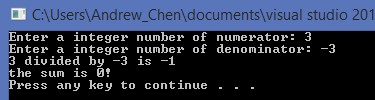
}

system("pause");

return 0;

}

Output:



If the inside catch function does not match the integer parameter, what will happened?

The answer is the program will keep trying to find a catch function below until the end of code (outside the try{} because the throw will break the try{}).

So it is ok to have a catch below the try.

Let’s see the example:

int main()

{

try{

int numerator, denominator;

cout << "Enter a integer number of numerator: ";

cin >> numerator;

cout << "Enter a integer number of denominator: ";

cin >> denominator;

if (denominator == 0) throw "denominator can not be 0!";

cout << numerator << " divided by " << denominator << " is "

<< numerator / double(denominator) << endl;

try

{

if (numerator + denominator == 0) **throw 0**;

cout << "The sum is: " << numerator + denominator << endl;

}

**catch (double n)**

**{**

**cout << "the sum is 0!" << endl;**

**}**

}

catch (char \*c)

{

cout << c << endl;

}

**catch (int n)**

**{**

**cout << "The sum is 0!" << endl;**

**}**

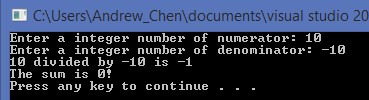
system("pause");

return 0;

}

Now I change the inside catch parameter to double(red color), it will not match the integer 0 I throw. But I have another catch match integer parameter below(green color), so the system will still find it and work

Output:



Notice if it is a nesting try function, throw will break the current try{}, and trying to find the catch below, but it can’t go above the try{}. So it will not work in this case:

int main()

{

try{

int numerator, denominator;

cout << "Enter a integer number of numerator: ";

cin >> numerator;

cout << "Enter a integer number of denominator: ";

cin >> denominator;

if (denominator == 0) **throw "denominator can not be 0!";**

cout << numerator << " divided by " << denominator << " is "

<< numerator / double(denominator) << endl;

try

{

if (numerator + denominator == 0) throw 0;

cout << "The sum is: " << numerator + denominator << endl;

}

**catch (char \*c)**

**{**

**cout << "haha" << c << endl;**

**}**

}

catch (int n)

{

cout << "the sum is 0!" << endl;

}

system("pause");

return 0;

}

Although There is a catch function match the cstring parameter type, but it is inside the try{}. When the first throw works, it breaks the first try{} which means everything inside the try{} will be ignored(be passed). System want to find a catch below the try{} but there is no catch below match cstring parameter, It finally fail.