1918
TALLINNA
TEHNIKAÜLIKOOL

DEBUGGING

Debugging I

- There are three debugging techniques:
- ***Breakpoints:*** Breakpoints are specific lines of code inside the source file where the debugger will suspend the execution and will give us an interactive prompt to mingle with the running process
- ***Watch:*** While you're debugging, you can use Watch windows to watch variables and expressions. Watch windows can display several variables at a time while debugging.
- ***Step:*** Stepping is the process of controlling step-by-step execution of the program. ***Step over:*** Steps over the current line of code and takes you to the next line. ***Step into:*** Steps into the method to show what happens inside it. Use this option when you are not sure the method is returning a correct result.

Debugging II

- **#define DEBUG**
-
- If you decide to define this variable then whatever is enclosed in the #ifdef / #endif blocks will be compiled. Otherwise not. You can define it by including #define DEBUG at the top (in the source code), or as a flag in the compile parameters. You don't have to physically alter the source file.

- *#ifdef DEBUG*
- *for(int j=0;j<=i;j++)printf("%3d",loos[j]);printf("\n");*
- *#endif*

Debugging III

- An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.
- Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.
- **throw** − A program throws an exception when a problem shows up. This is done using a throw keyword.
- **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.
- **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.

Debugging IV

- Debugging example: function ‘minutes’ that takes a string in a format: “hh:mm” as a parameeter and returns an integer (parameeter value converted to minutes). Function catches 3 different exceptions and throws a different error code for each:
- 1 – more than 23 hours;
- 2 – more than 59 minutes;
- 3 – missing a ‘:’.

Debugging V

```
int minutes(char *p){
    int h, m;
    char *pos;
    h=atoi(p); // get hours from string p
    if (h>23) throw 1; // too many hours
    pos=strchr(p,':'); // find ‘:’
    m=(pos==NULL)?-1:atoi(pos+1);
    if (m>59) throw 2; // too many minutes
    if (m<0) throw 3; // missing ‘:’
    return h*60+m;
}
```

Debug VI

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

int main(){
    int amount;
    cout << „Insert a number: ";
    cin >> amount;
    if(amount>=10){ throw „Too high number";}
    cout << amount << endl;
    //system("pause");
    return(0);
```

Debug VII

```
#include <iostream>
using namespace std;
int main(){
   int amount;
   try{
    cout << „Insert a number: ";
    cin >> amount;
    if(amount>=10) throw 1;
    cout << amount << endl;
   } catch(int nr){
     cout << „Error " << nr << endl;
   }
   //system("pause");
   return(0);
}
```

Debug VIII

```
#include <iostream>
#include <stdexcept>
using namespace std;
class SizeError : public runtime_error { //error subclass
public:
  SizeError(const string& serror = "") : runtime_error(serror) {}
};
int main() {
  try {
    int amount;
    cout << „Insert a number: ";
    cin >> amount;
    if(amount>=10)throw SizeError(„Too high number");
    cout << amount << endl;
  }
  catch (SizeError& v) {
    cout << v.what() << endl;
  }//system("pause");
  return(0);
}
```

Debugging IX

```
#include <iostream>
#include <stdexcept>
using namespace std;
class SizeError : public runtime_error { //error subclass
public:
  SizeError(const string& serror = "") : runtime_error(serror) {}
};
int readNR(){
    int amount;
    cout << „Insert a number: ";
    cin amount;
    if(amount>=10)throw SizeError(„Too high number");
    return amount;}
int main() {
  try {
    int a=readNR();
    cout << a << endl;
  }
  catch (SizeError& v) {
    cout << v.what() << endl;
  } //system("pause");
  return(0);
}
```

Questions

- 1.Explain the output of codes in slides „Debugging VI-IX“ when inputting 50. 2x2 points

- 2.Read an array of dates from a file(use classes and exceptions in your code), sort all dates into 3 files (EU format, US format, nonformatted)
- *(if a date fits into both formats, use EU format file)*