

Искусство отладки

Алексей Дрожжин



Структура лекции

Основные разделы и подразделы.

- ❑ Введение
- ❑ Подготовка и планирование
 - Неизбежность отладки,
 - Поддержка процессом,
 - Информационное и программное обеспечение
- ❑ Инструменты отладки ПО
 - Debugger,
 - Profiler,
 - Memory leaks detector,
 - UT Coverage
- ❑ Отладчик Visual Studio
 - Break points
 - Watch window
- ❑ Типовые дефекты
- ❑ Crash dumps
- ❑ Q&A
- ❑ Литература

Введение

Появление термина debugging.

Bug :

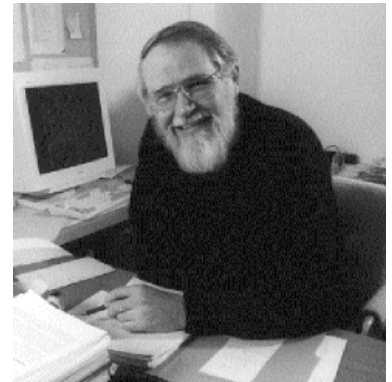
- Клоп
- Букашка
- Жук
- Микроб



Введение: Определение отладки (“debugging”)

Процесс определения и устранения причин ошибок в программе.

“ Отлаживать код вдвое сложнее, чем писать.
Если Вы используете весь свой интеллект
при написании программы, вы по определению
не достаточно умны, чтобы её отладить. ”



Брайан Керниган



Введение: Эффективность навыка отладки

Исследования далёкого 1975 года.

Отобрали группу профессиональных программистов с 4-х летним стажем: 3 - “лучших”, 3 - “худших”.

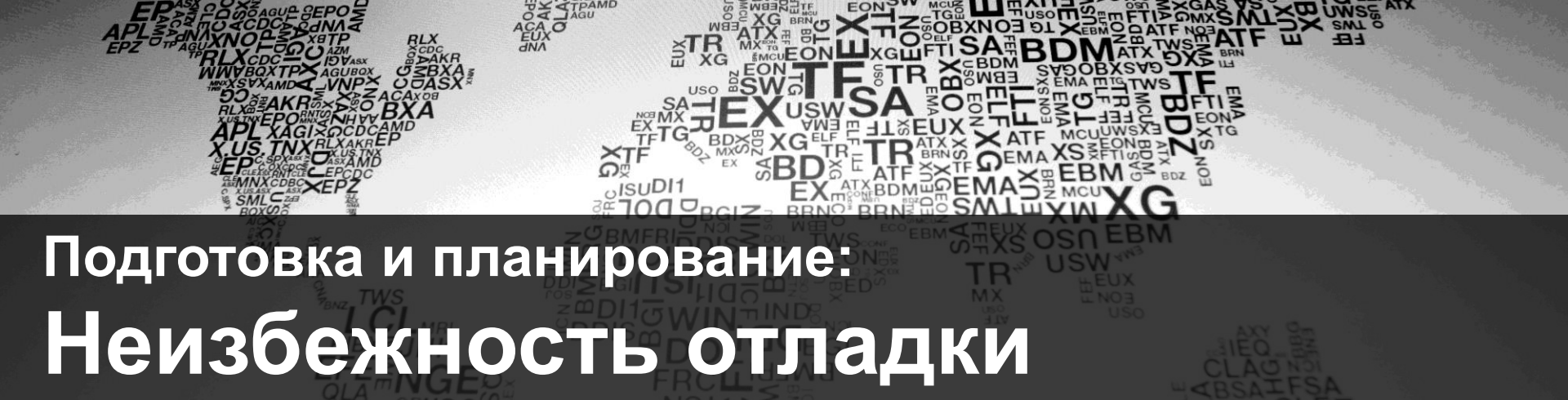
Задача: найти и исправить программу, содержащую 12 дефектов.

	“Лучший”	“Худший”
Среднее время отладки (мин.)	5,0	14,1
Число не обнаруженных дефектов	0,7	1,7
Число внесенных дефектов	3,0	7,7

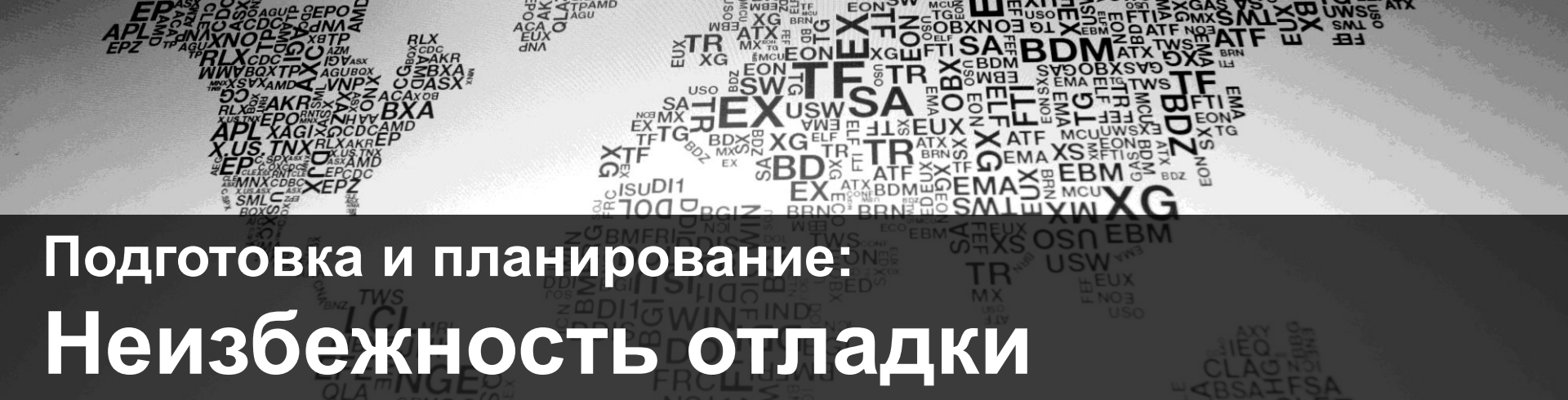


Подготовка и планирование

- Неизбежность отладки,
- Поддержка процессом,
- Информационное и программное обеспечение
- Алгоритм поиска и устранения дефекта



Подготовка и планирование: Неизбежность отладки



Подготовка и планирование: Неизбежность отладки

- Человеческий фактор,



Подготовка и планирование: Неизбежность отладки

- Человеческий фактор,
- Короткие/невозможные сроки,



Подготовка и планирование: Неизбежность отладки

- Человеческий фактор,
- Короткие/невозможные сроки,
- Непонимание требований,



Подготовка и планирование: Неизбежность отладки

- Человеческий фактор,
- Короткие/невозможные сроки,
- Непонимание требований,
- Недостаток экспертизы/знаний,



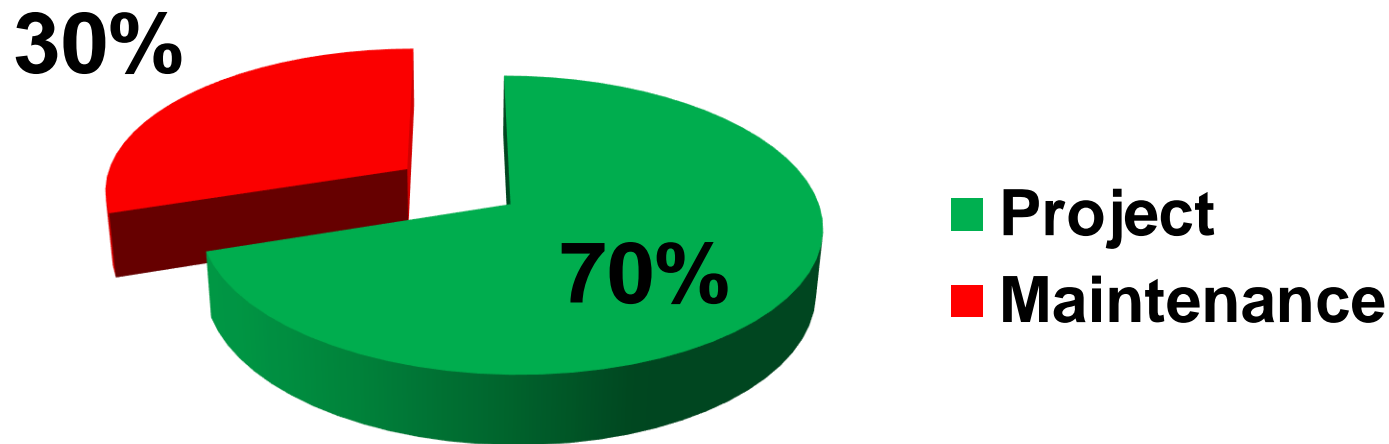
Подготовка и планирование: Неизбежность отладки

- Человеческий фактор,
- Короткие/невозможные сроки,
- Непонимание требований,
- Недостаток экспертизы/знаний,
- Пренебрежение качеством.



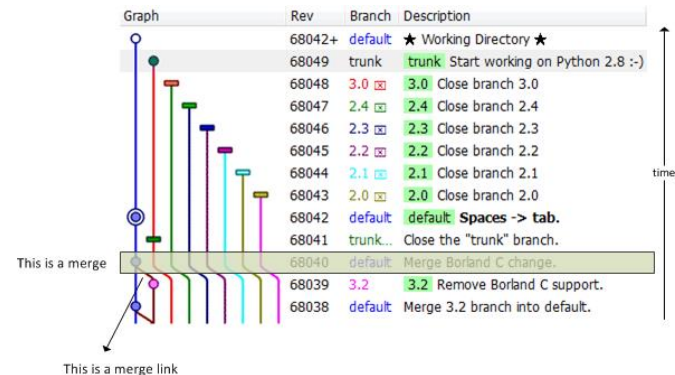
Подготовка и планирование: Поддержка процессом

До 20-40% ресурсов тратится на поддержку существующего ПО, поиск и исправление дефектов.



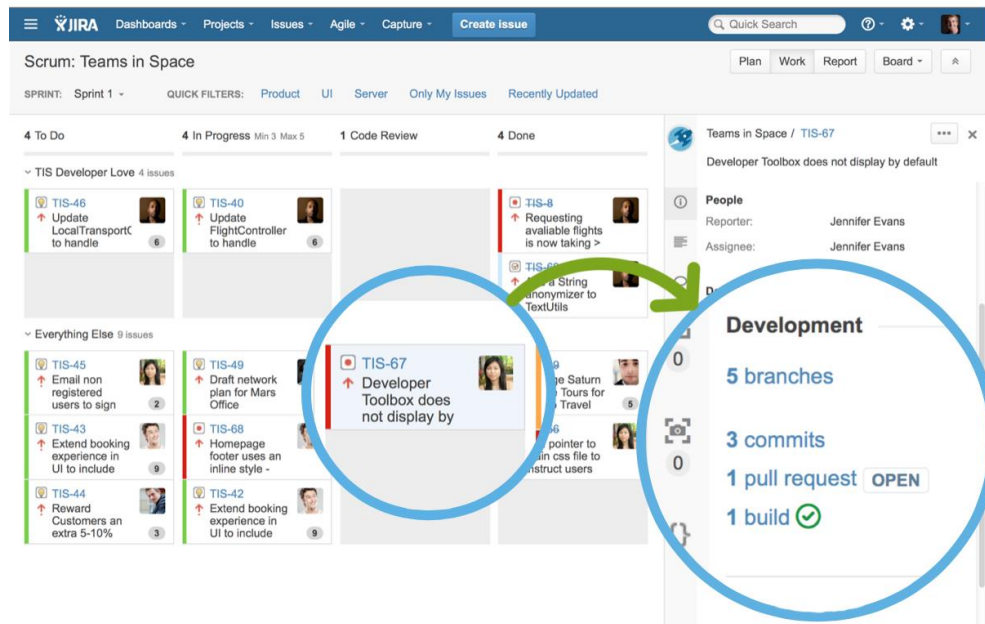
Подготовка и планирование: Информационное и программное обеспечение

Системы контроля версий.



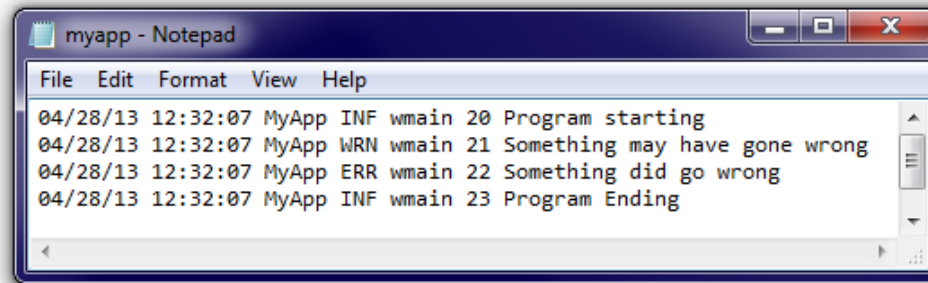
Подготовка и планирование: Информационное и программное обеспечение

Системы отслеживания дефектов.



Подготовка и планирование: Информационное и программное обеспечение

Логирование.



Возможности хорошего логгера:

- Уровни логирования и фильтрация сообщений
- Ротация лог-файлов
- Возможность записи сообщений не только в файлы
- Потокбезопасность
- Асинхронное логирование
- Гибкое форматирование и конфигурация логов

Подготовка и планирование: Информационное и программное обеспечение

Уровни логгирования.



Debug - сообщения отладки, профилирования



Information - обычные сообщения, информирующие о действиях системы.



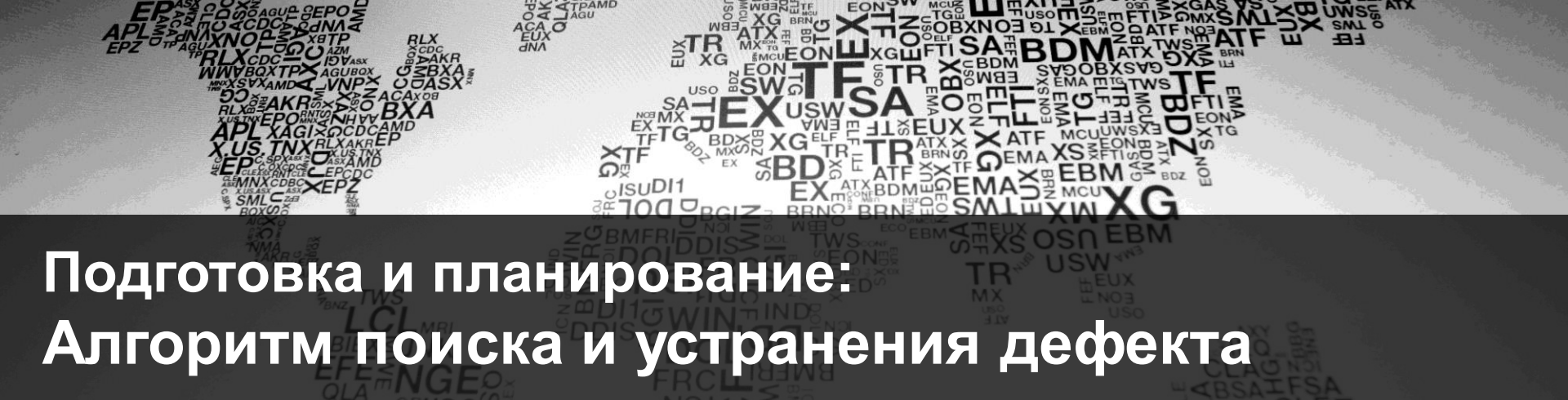
Warning - произошло что-то странное. Следует разобраться в том, что произошло, что это означает, и отнести ситуацию либо к инфо-сообщению, либо к ошибке



Error - ошибка в работе системы, требующая вмешательства. Что-то не сохранилось, что-то отвалилось и т.д.



Fatal - особый класс ошибок. Такие ошибки приводят к неработоспособности системы в целом, или неработоспособности одной из подсистем.



Подготовка и планирование: Алгоритм поиска и устранения дефекта

Основные шаги:

Воспроизведение (reproducing)

Поиск бага (investigation)

Поиск решения (fixing)

Проверка решения (testing)



Подготовка и планирование: Краткий повтор раздела

- Неизбежность отладки,
 - человеческий фактор,
 - сроки,
 - требования,
 - знания,
 - качество.
- Поддержка процессом,
- Информационное и программное обеспечение
 - Системы контроля версий (git, mercurial, svn, cvs,...)
 - Системы отслеживания дефектов
 - Логирование
- Алгоритм поиска и устранения дефекта



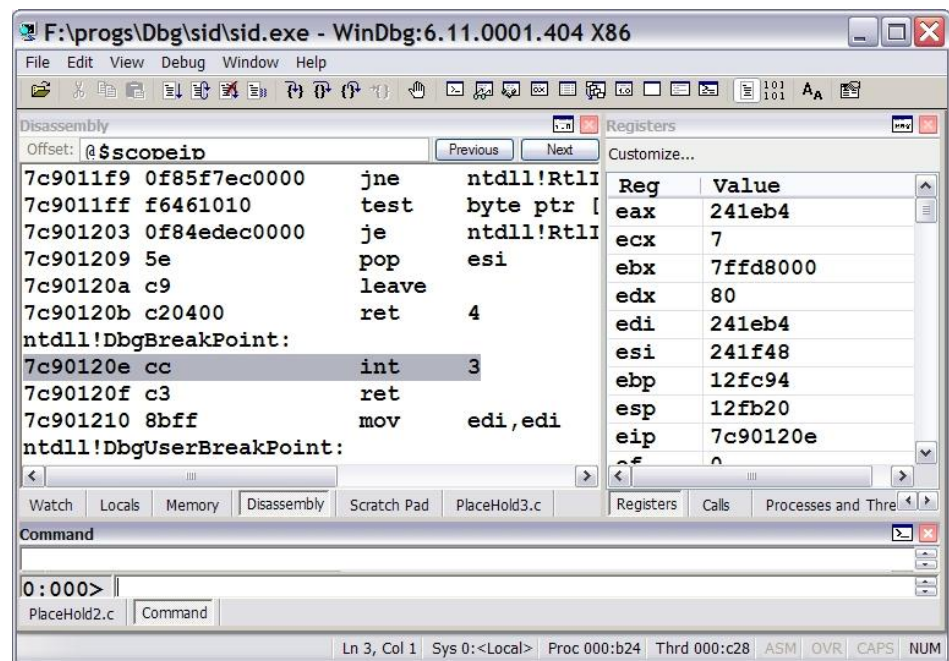
Инструменты отладки ПО

- Debugger,
- Profiler,
- Memory leaks detector,
- Testing

Инструменты отладки ПО: Debugger (отладчик)

Возможности:

- Пошаговая трассировка
- Отслеживание/установка /изменение значения переменных
- Установка/удаление условий остановки



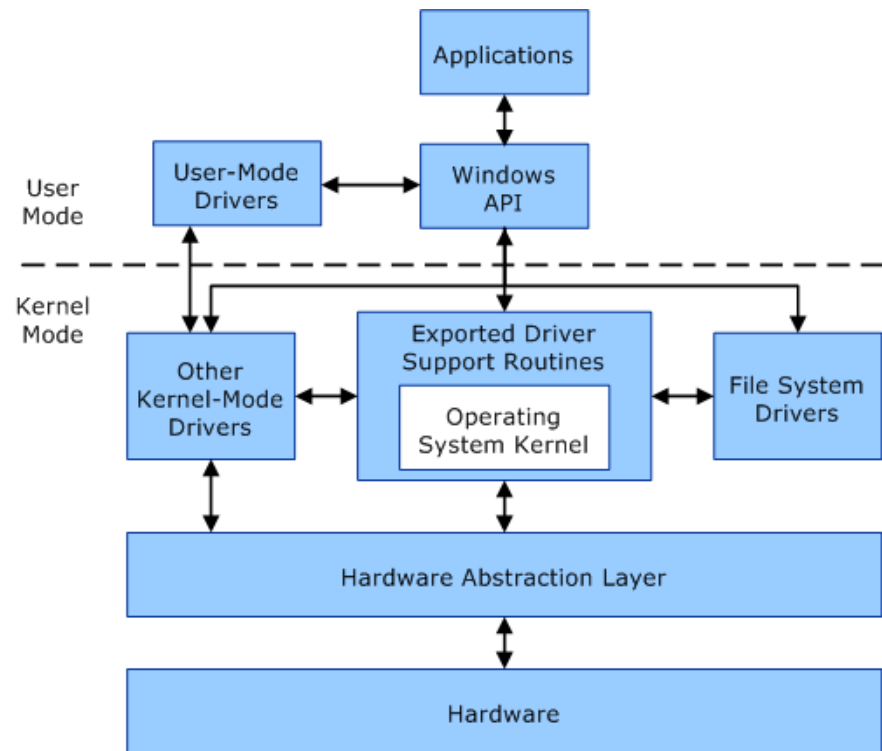
Инструменты отладки ПО: Debugger (отладчик)

Типы Windows-отладчиков

- Отладка в режиме пользователя
- Отладка в режиме ядра

Представители:

- Visual Studio
- WindDbg
- SoftICE
- Aqtime
- Dtrace
- IDA



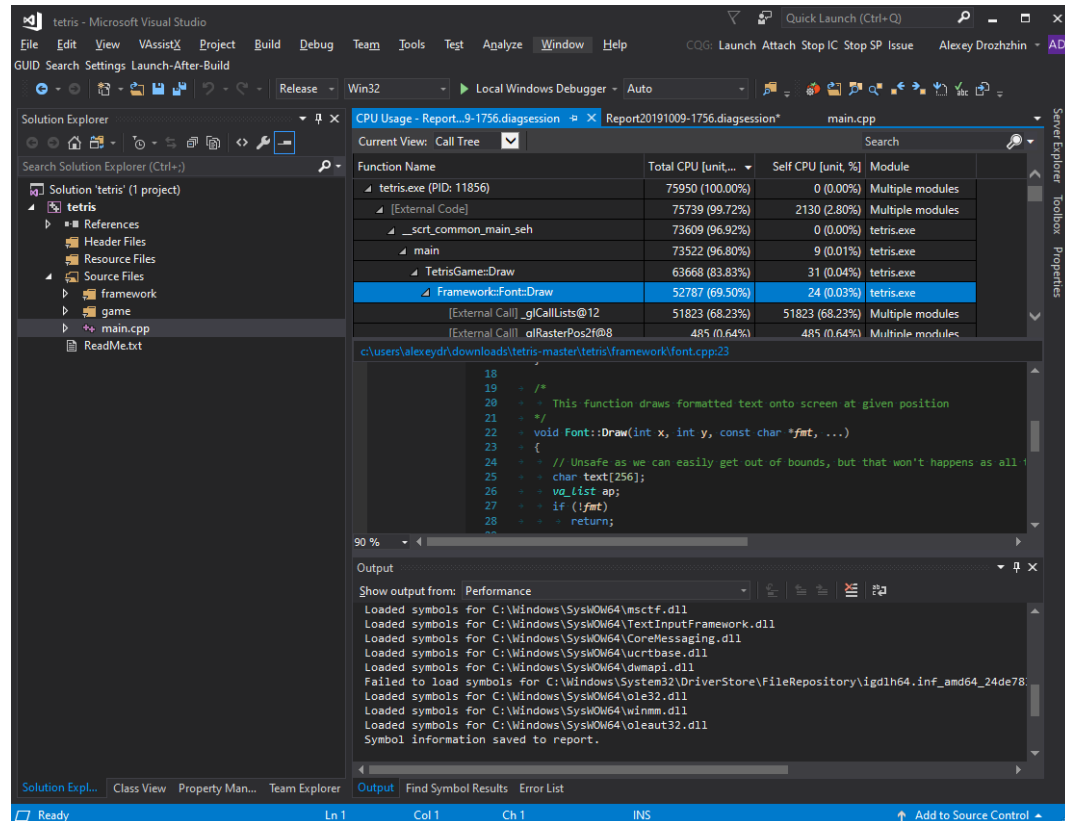
Инструменты отладки ПО: Profiler (профилировщик)

Определяет:

- время выполнения отдельных фрагментов,
- число верно предсказанных условных переходов,
- количества вызовов той или иной точки программы.

Представители:

- Intel Vtune,
- Visual Studio,
- CodeAnalyst,
- Aqtime,
- Valgrind



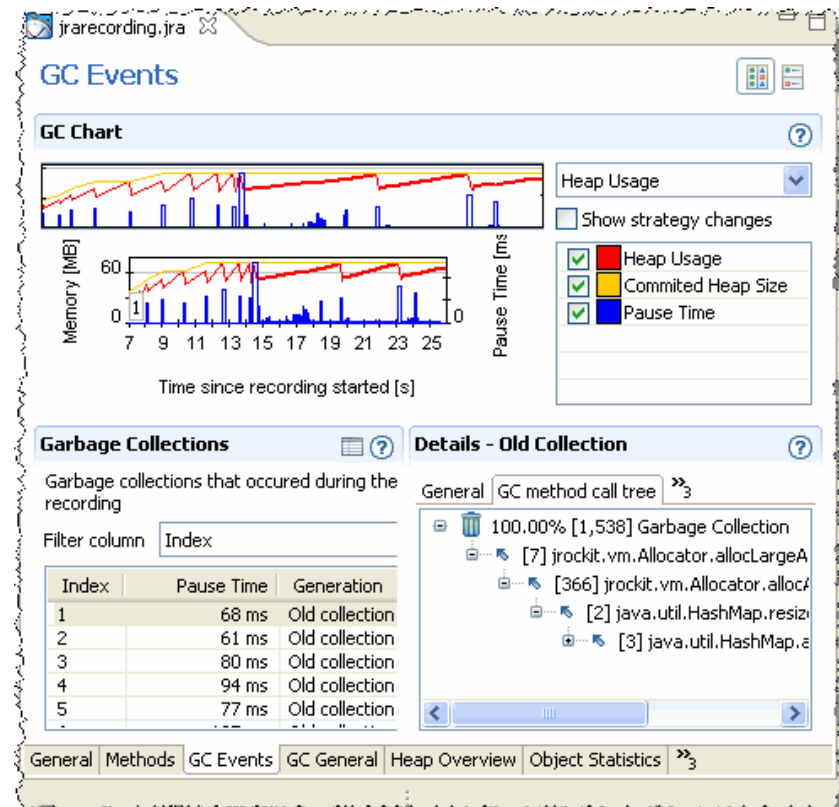
Инструменты отладки ПО: Memory leaks detector

Возможности:

- Выявление “утечки” памяти
- Анализ использования памяти

Представители:

- Visual Studio
- Visual Leak Detector
- Valgrind
- Glow Code

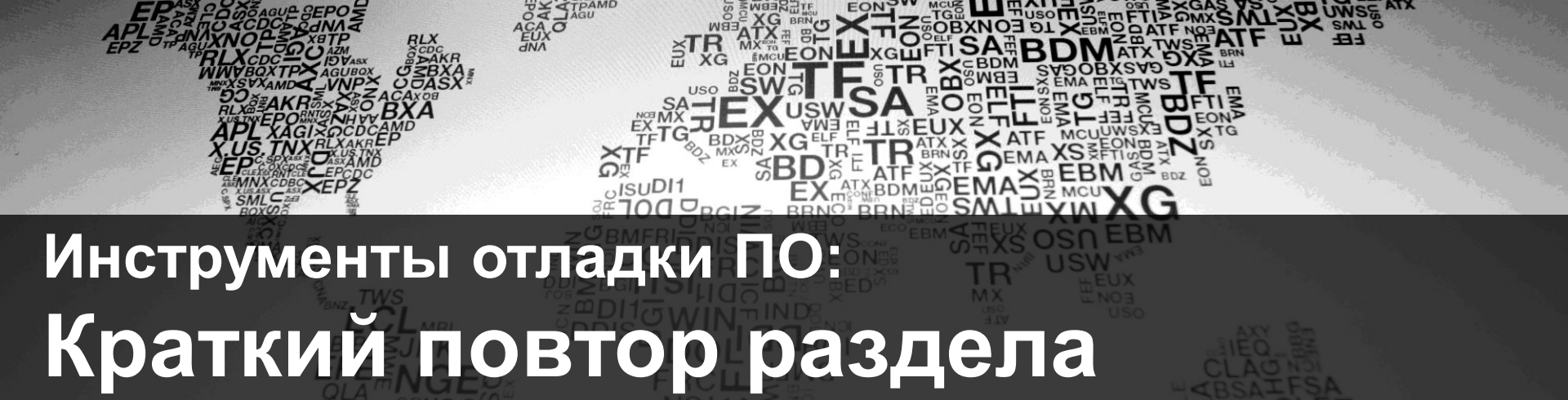




Инструменты отладки ПО: Testing

Применение:

- *Юнит Тестирование* — тестирование отдельных компонентов системы, чаще программно;
- *Нагрузочное Тестирование* — тестирование производительности системы;
- *Стресс Тестирование* — тестирование отказоустойчивости системы в нештатных ситуациях;
- *Интеграционное Тестирование* — комплексное тестирование системы после соединения всех отдельных компонентов;
- *Регрессивное Тестирование* — тестирование уже протестированных участков исходного кода.



Инструменты отладки ПО: Краткий повтор раздела

- Debugger,
- Profiler,
- Memory leaks detector,
- Testing



Отладчик Visual Studio

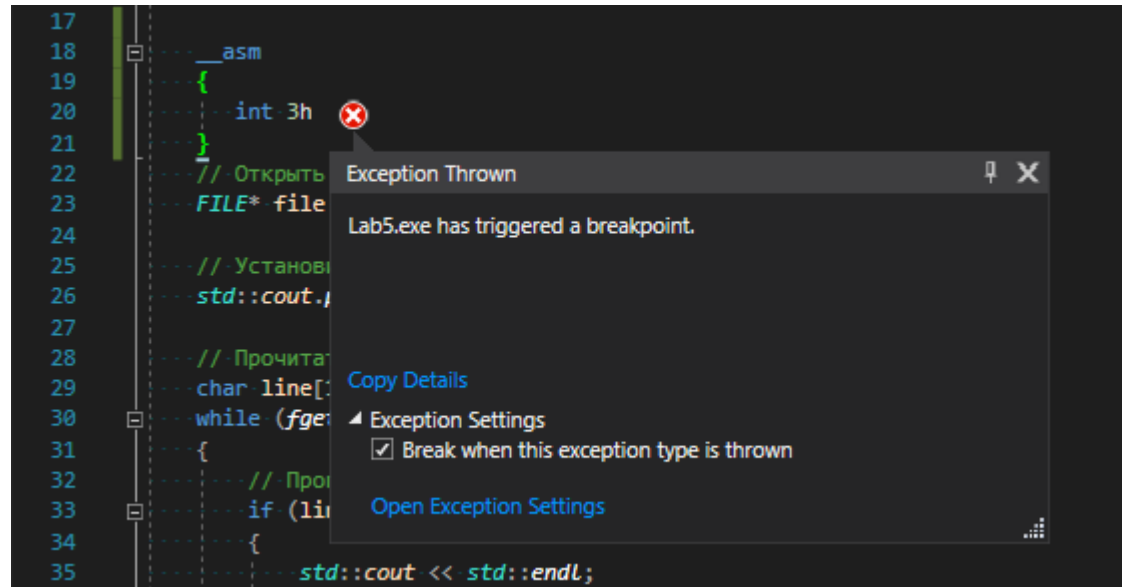
- Break points
- Watch window

Отладчик Visual Studio: Break points

```
13 int main(int argc, char* argv[])
14 {
15     // Прочитать имя исходного файла
16     char* inputFileName = argv[0];
17
18     // Открыть файл
19     FILE* file = fopen(inputFileName, "r");
20
21     // Установить точность вывода
```

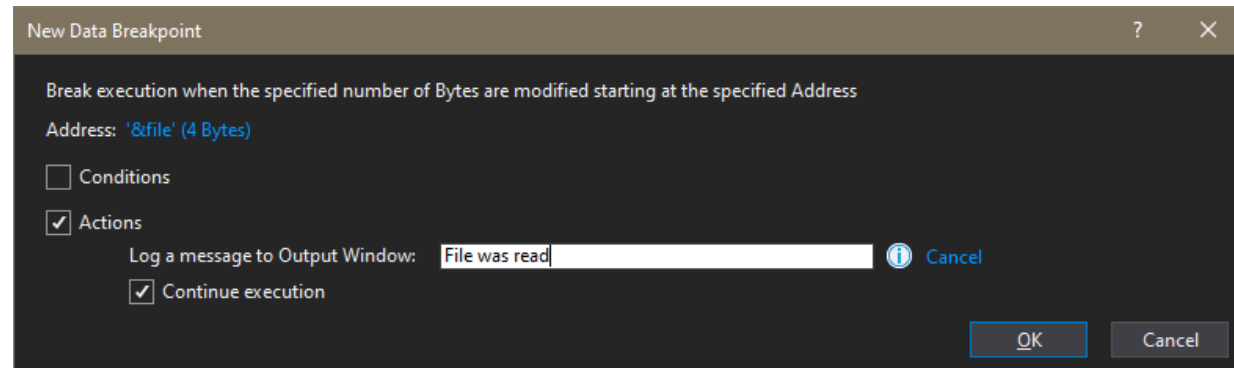
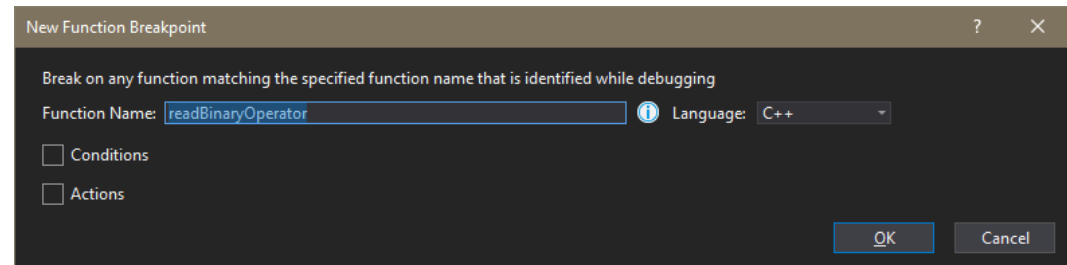
```
// STATUS_BREAKPOINT
// (0x80000003)
```

```
_asm
{
    int 3;
}
```



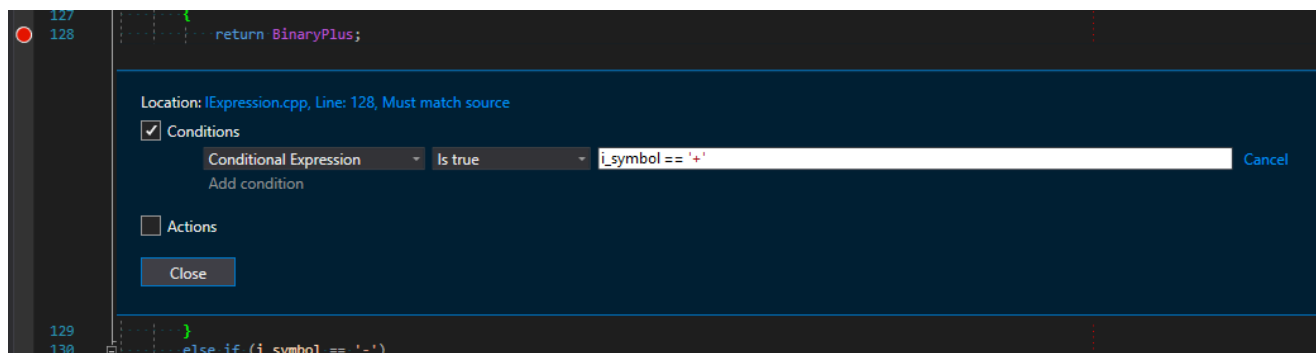
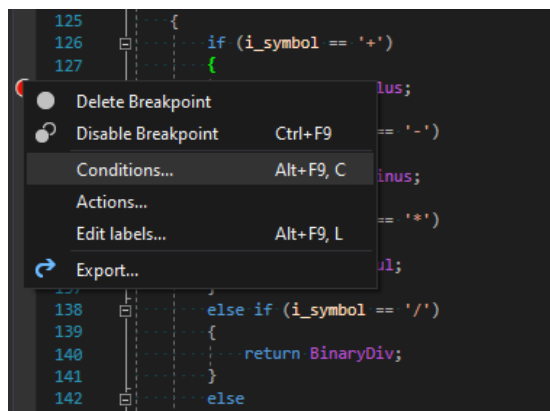
Отладчик Visual Studio: Break points: основные типы

- Simple breakpoint
- Break at function
- Data breakpoint



Отладчик Visual Studio: Break points: расширенные

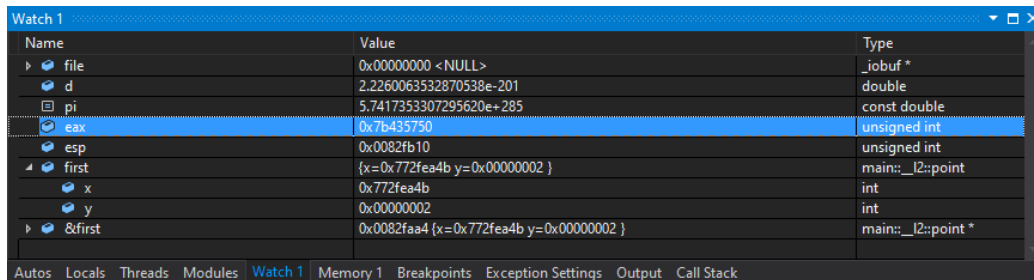
- С условием
(Condition...)
- По числу попаданий
(Hit Count...)
- Фильтр
(Filter...)
- По событию
(When Hit...)



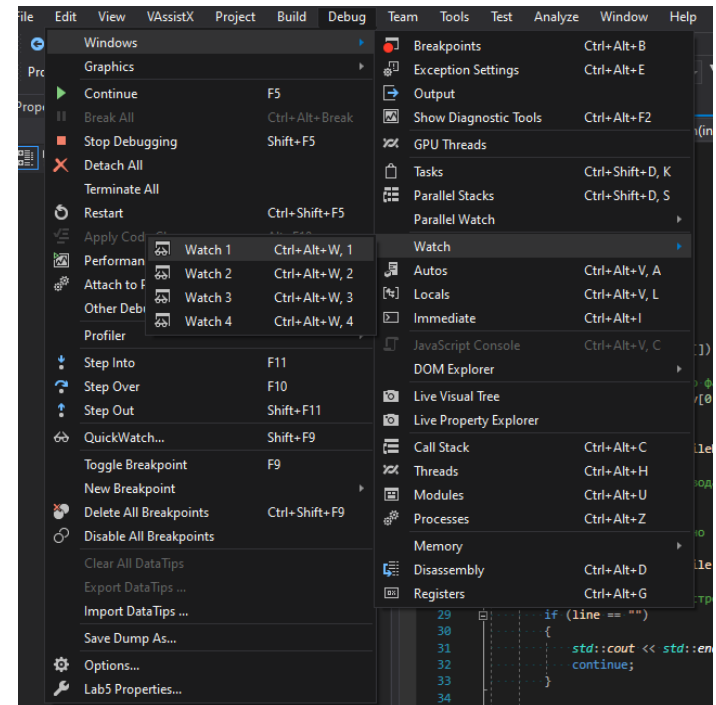
Отладчик Visual Studio: Watch window


Возможности:

- Просмотр значений переменных
- Редактирование переменных
- Просмотр псевдорегистров (@ERR, @EAX, @EBX, @ECX, ...)
- Форматированный просмотр (d, l, u, o, x, e, c, s, hr, wm)



Name	Value	Type
file	0x00000000 <NULL>	jobuf *
d	2.2260063532870538e-201	double
pi	5.7417353307295620e+285	const double
eax	0x7b435750	unsigned int
esp	0x0082fb10	unsigned int
first	{x=0x772fea4b y=0x00000002 }	main::_l2::point
x	0x772fea4b	int
y	0x00000002	int
&first	0x0082faa4 {x=0x772fea4b y=0x00000002 }	main::_l2::point *





Отладчик Visual Studio: Краткий повтор раздела

- Break points
- Watch window



Типовые дефекты

- Народная мудрость
- Типы дефектов
 - Access violation
 - Memory leak
 - Heap corruption
 - Stack overflow
 - Deadlocks



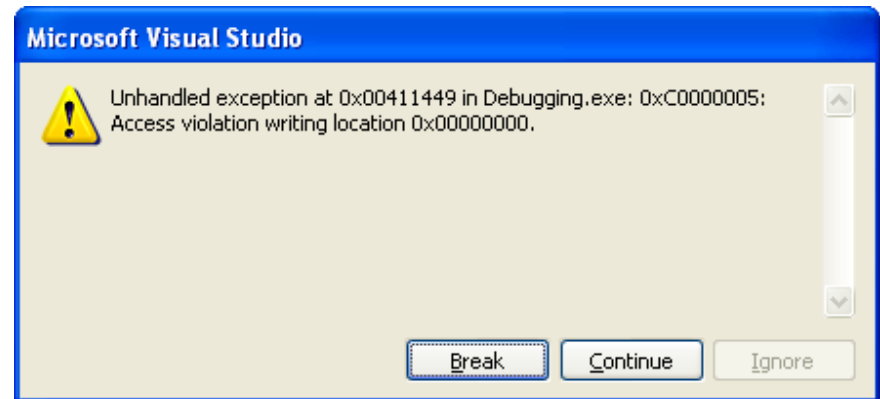
Типовые дефекты: Народная мудрость

Обычно это так:

- Ошибка в собственном коде.
- Ошибка имеет простое решение.
- Ошибка возникла в последнем изменении кода.
- Сегодня ошибку исправить проще, чем завтра.

Типовые дефекты: Типы дефектов

- Access violation
- Memory leak
- Heap corruption
- Stack overflow
- Deadlocks

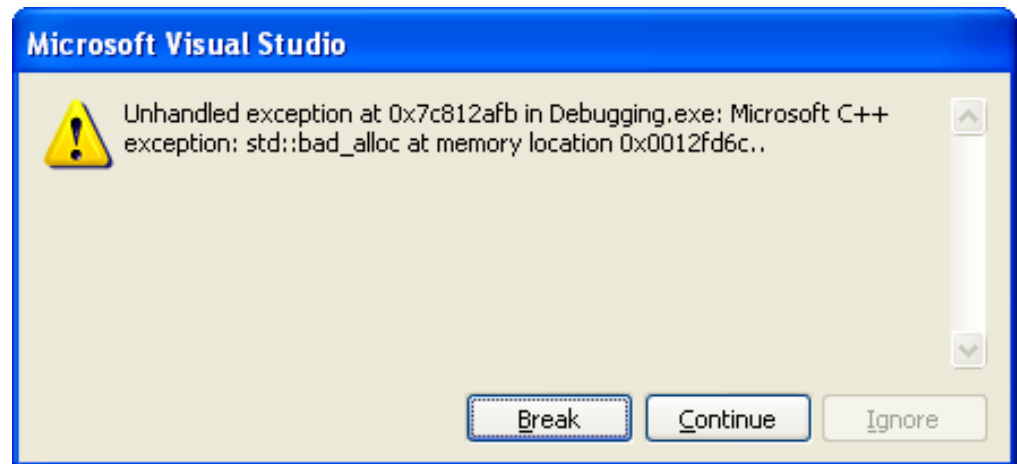


Причины:

- Неинициализированные переменные и члены класса
- Осиротевшие указатели

Типовые дефекты: Типы дефектов

- Access violation
- Memory leak
- Heap corruption
- Stack overflow
- Deadlocks

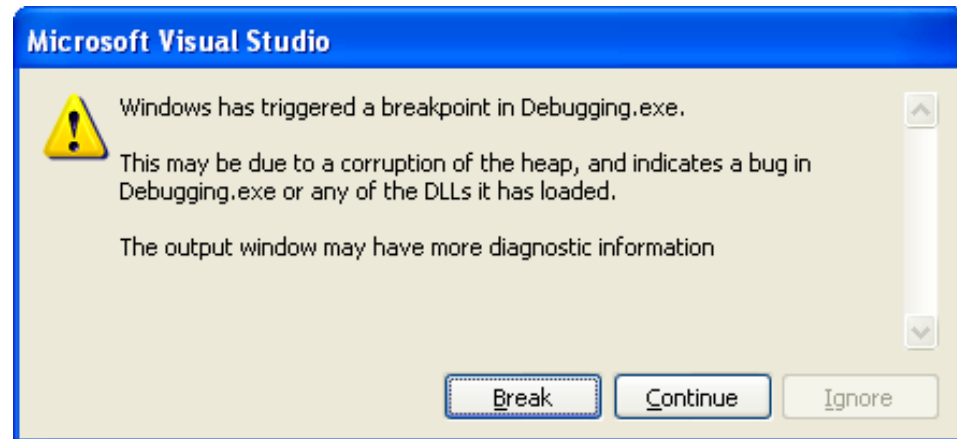


Причины:

- Забытый delete
- Не виртуальный деструктор в базовом классе

Типовые дефекты: Типы дефектов

- Access violation
- Memory leak
- Heap corruption
- Stack overflow
- Deadlocks

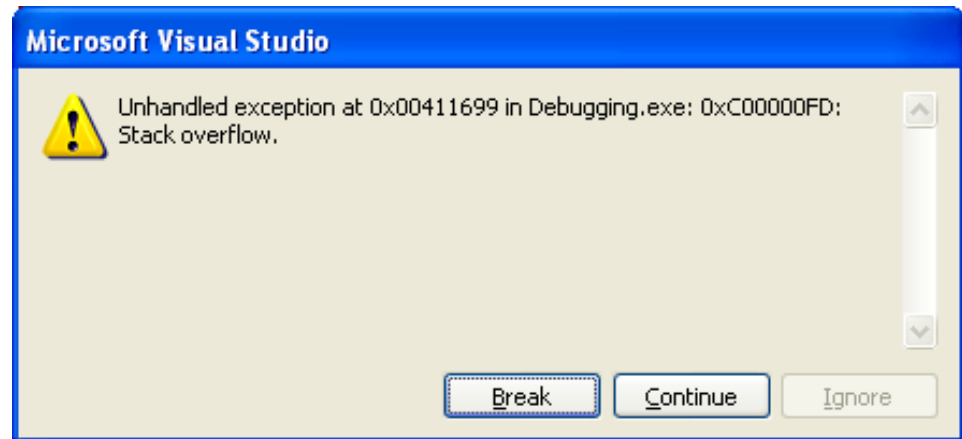


Причины:

- Использование нескольких указателей на один адрес в куче
- Не корректное приведение типов

Типовые дефекты: Типы дефектов

- Access violation
- Memory leak
- Heap corruption
- Stack overflow
- Deadlocks

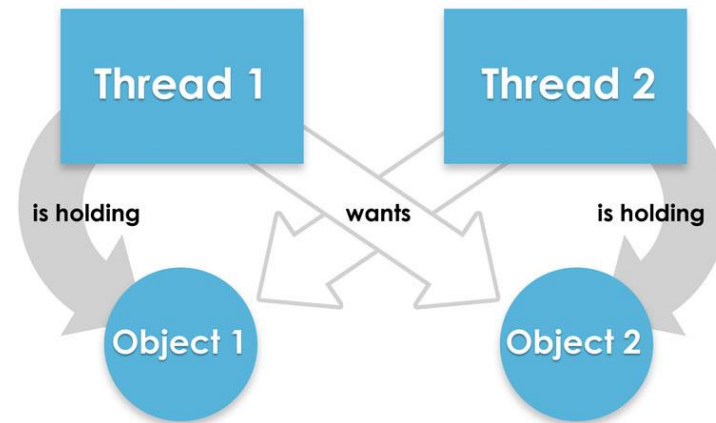


Причины:

- “Бесконечная” рекурсия

Типовые дефекты: Типы дефектов

- Access violation
- Memory leak
- Heap corruption
- Stack overflow
- Deadlocks



Причины:

- Не правильная синхронизация доступа к объектам.



Типовые дефекты: Краткий повтор раздела

- Народная мудрость
- Типы дефектов
 - Access violation
 - Memory leak
 - Heap corruption
 - Stack overflow
 - Deadlocks

Crash dump





Crash dump

- Назначение
- Создание
- Анализ
 - Без PDB
 - С PDB



Crash dump: Назначение

Содержит информацию о состоянии программы в определённый момент времени:

- Снимок памяти
- Поток приложения
 - Стек вызовов
 - Значения регистров ЦП
 - Значения локальных переменных
- Информация об ошибке/исключении



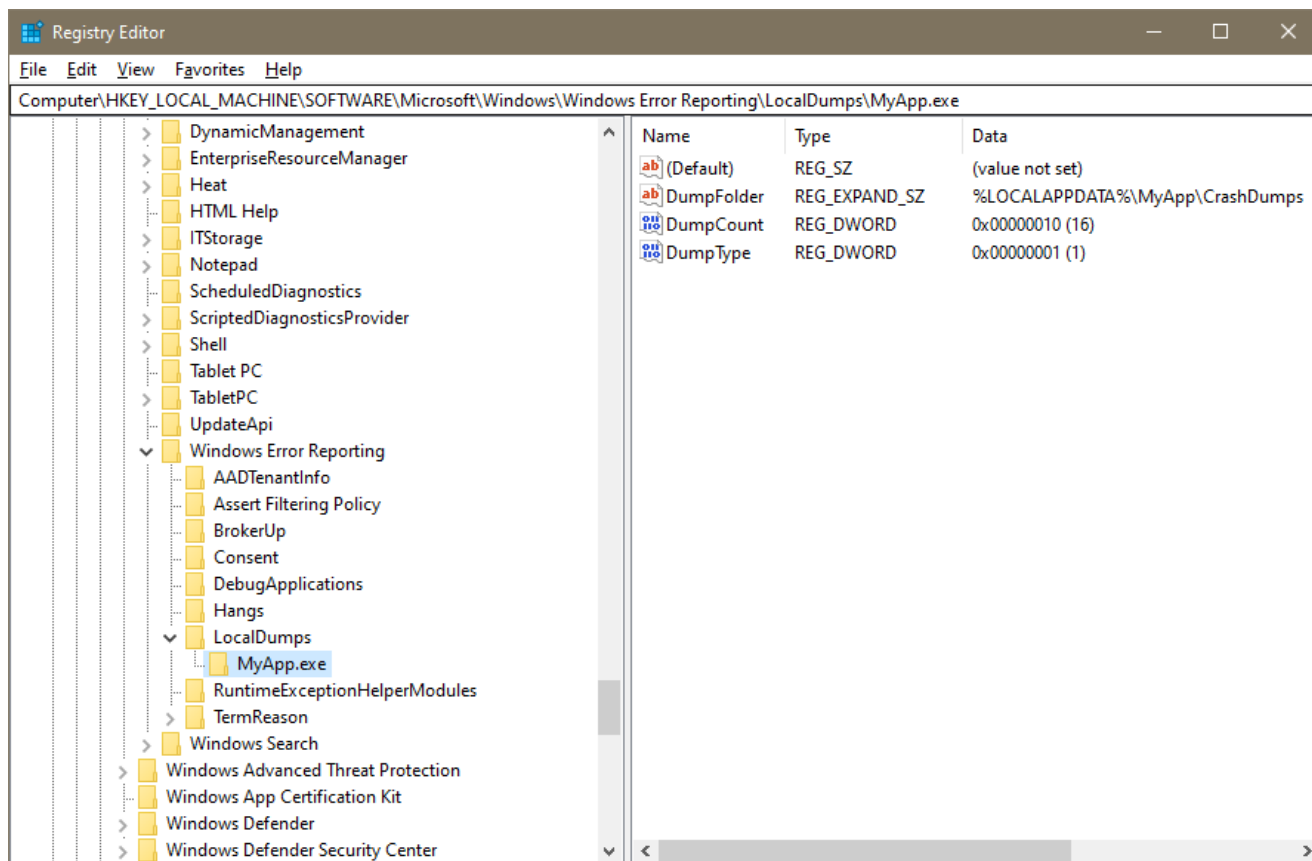
Crash dump: Создание (вариант 1)

```
#include <dbghelp.h>
```

```
BOOL WINAPI MiniDumpWriteDump( HANDLE hProcess,  
    DWORD ProcessId,  
    HANDLE hFile,  
    MINIDUMP_TYPE DumpType,  
    PMINIDUMP_EXCEPTION_INFORMATION ExceptionParam,  
    PMINIDUMP_USER_STREAM_INFORMATION UserStreamParam,  
    PMINIDUMP_CALLBACK_INFORMATION CallbackParam );
```

```
#include <windows.h>  
LONG CustomTopLevelFilter(_EXCEPTION_POINTERS *pExceptionInfo )  
{  
    return GenerateDump(pExceptionInfo);  
}  
void SetExceptionHook()  
{  
    ::SetUnhandledExceptionFilter(TopLevelFilter );  
}
```


Crash dump: Создание (вариант 2)



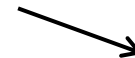
Crash dump: Анализ

WinDbg



v1.0-20120505-122822-5836-7776.dmp
v1.0-20120505-122943-5600-4628.dmp
v1.0-20120505-123017-2680-6400.dmp

Visual Studio



```
Microsoft (R) Windows Debugger Version 6.12.0002.633 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

Loading Dump File [C:\Users\romangol\AppData\Local\Temp\AppName\v1.0-20120505-123017-2680-6400.dmp]
User Mini Dump File: Only registers, stack and portions of memory are available

Symbol search path is: *** Invalid ***
*****
Symbol loading may be unreliable without a symbol search path.
* Use .symfix to have the debugger choose a symbol path.
* After setting your symbol path, use .reload to refresh symbol locations.
*****
Executable search path is:
Windows 7 Version 7601 (Service Pack 1) MP (4 procs) Free x86 compatible
Product: WinNt, suite: SingleUserTS
Machine Name:
Debug session time: Sat May 5 12:30:58.000 2012 (UTC + 4:00)
System Uptime: not available
Process Uptime: 0 days 0:00:53.000

This dump file has an exception of interest stored in it.
The stored exception information can be accessed via .ecxr.
(a781900): Integer divide-by-zero - code c0000004 (first/second chance not available)
eax=00000000 ebx=00380d78 ecx=00000000 edx=00000000 esi=00380d38 edi=0017ebd8
eip=77e40c22 esp=0017e898 ebp=0017e8a8 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for ntdll.dll -
ntdll!ZwGetContextThread+0x12:
77e40c22 83c404          add     esp,4
```

Minidump File Summary
10/11/2019 12:08:38 PM

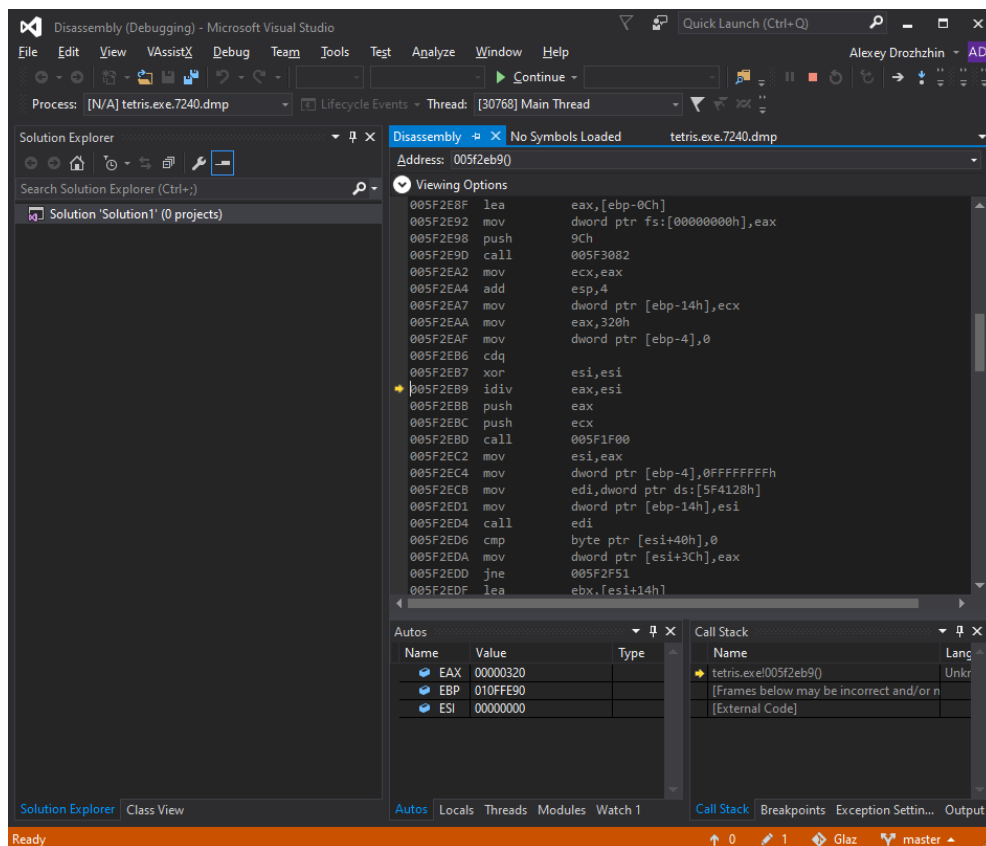
Dump Summary	
Dump File	tetris.exe.7240.dmp : D:\dumps\tetris\CrashDumps\tetris.exe.7240.dmp
Last Write Time	10/11/2019 12:08:38 PM
Process Name	tetris.exe : C:\Users\alexeid\Downloads\tetris-master\Release\tetris.exe
Process Architecture	x86
Exception Code	0xC0000094
Exception Information	The thread tried to divide an integer value by an integer divisor of zero.
Heap Information	Not Present
Error Information	

System Information	
OS Version	10.0.18362
CLR Version(s)	

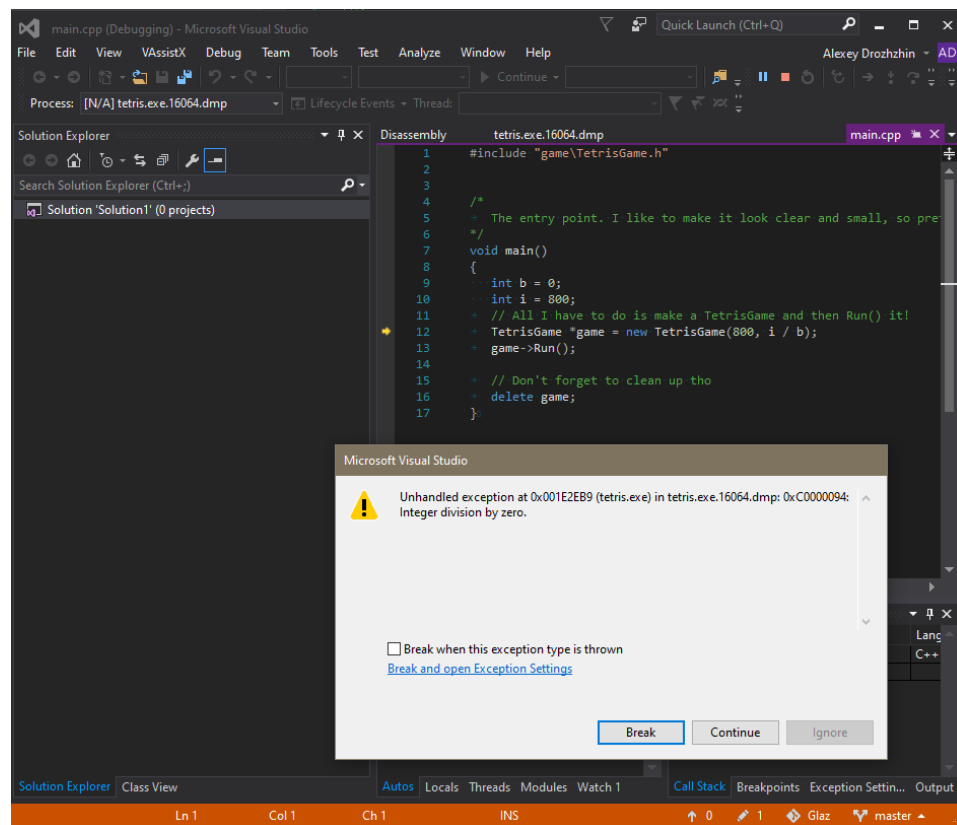
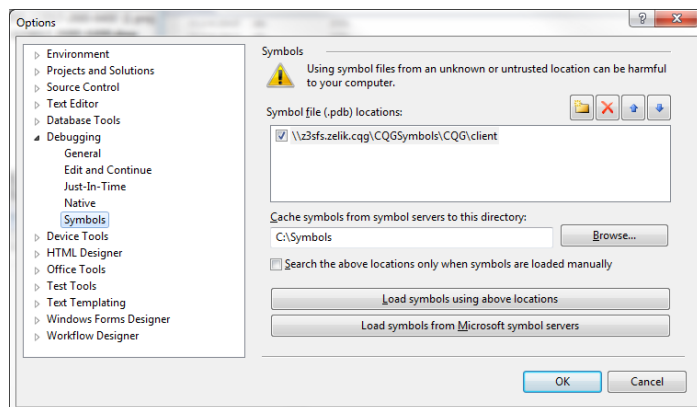
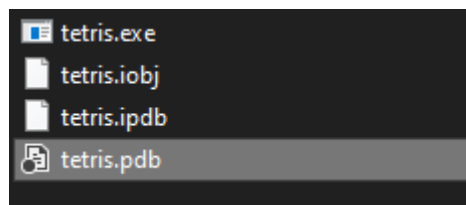
Modules	
Module Name	Module Path
tetris.exe	C:\Users\alexeid\Downloads\tetris-master\Release\tetris.exe
ntdll.dll	C:\Windows\System32\ntdll.dll
kernel32.dll	C:\Windows\System32\kernel32.dll

Output
Show output from: General
We were unable to automatically populate your Visual Studio Team Services accounts.
The following error was encountered: VS30063: You are not authorized to access https://app.vssps.visualstudio.com.

Crash dump: Анализ : без PDB



Crash dump: Анализ : с PDB





Crash dump: Краткий обзор материала

- Назначение
- Создание
- Анализ
 - Без PDB
 - С PDB



Q&A

Очень интересная литература



- Tarik Soulati, Inside Windows Debugging: A Practical Guide to Debugging and Tracing Strategies in Windows
- Д.Роббинс. Поиск и устранение ошибок в программах под Windows
- С.Макконнелл. Совершенный код
- Д.Востоков, Memory Dump Analysis Anthology



1 800-525-7082

www.cqg.com

Disclaimer

Trading and investment carry a high level of risk, and CQG, Inc. does not make any recommendations for buying or selling any financial instruments. We offer educational information on ways to use our sophisticated CQG trading tools, but it is up to our customers and other readers to make their own trading and investment decisions or to consult with a registered investment advisor.

© 2019 CQG, Inc. All rights reserved.

CQG®, DOMTrader®, SnapTrader®, TFlow®, TFOBv®, TFOBVO®, TFlow®, and Data Factory™ are trademarks of CQG, Inc.