Why VC++ instead of Dev C++?

- I love UNIX! I am proficient in UNIX!
- I like public domain open source software.
- I love GPL.
- I was more confident in GCC than in Microsoft C.
- But! The software business has changed so much since year 1990. One man heroic software is no more astonishing and reliable.
- Instead, software engineers together contribute to the functionalities of a package.

- Dev-C++ 5.0 beta 9.2 (4.9.9.2) 2005-02-21 is a Win32 porting (Mingw) of GNU GCC 3.4.2
- http://www.bloodshed.net/devcpp.html
- http://www.bloodshed.net/dev/devcpp.html

Problem #1

```
#include <stdio.h>
int main (void)
{
   char *filename;

   printf ("Please input the filename of the data. (ex: "
           " pixelsImage1.dat)\n\n >> ");
   scanf ("%s", filename);
   return 0;
}
```

This does not cause any runtime error!!

Is it only too lucky for this or OK to use every unallocated memory?

Problem #2

```
int fun(int size)
{
   int a[size];
   ...
}

int s = -10;
fun(s);
```

https://gcc.gnu.org/onlinedocs/gcc/Variable-Length.html

Variable Length Array (VLA) of ISO C99

Since a is allocated on the system stack, this can be implemented. However, at least some dynamic check of array size is necessary. malloc() has return value but VLA does not.

...

Could you imagine that this is not a compile time error and even not a runtime error in some context?
I verified that GCC 3.4 accepts this but VC series do not.
Poor portability, not supported in ANSI C++

Problem #3

```
#include <iostream>
using namespace std;
int main()
{
   char hexnum[8];
   char temp[8];
   cin >> hexnum;
   temp = hexnum;
   cout << temp;
   return 0;
}
```

VC error C2106: '=' : left operand must be l-value

g++ 4.8.3
error: invalid array assignment
But no error in g++ 3.4.2

Can you imagine that this is allowed in Dev C++ 4.9.9.2?

Problem #4

- Tolerate “missing return statements”

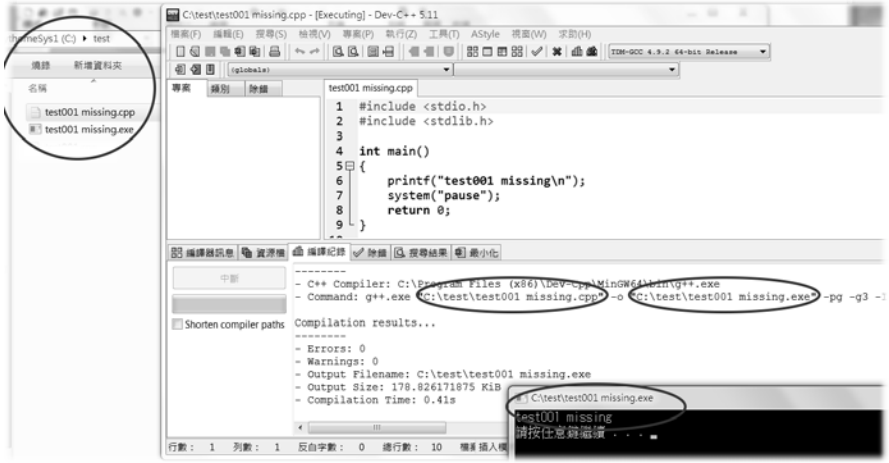
```
#include <stdio.h>
int square(int x) {
   int z = x * x;
   int y;
   y = 10*x+1;
}
int main() {
   printf("%d\n", square(10));
}
```

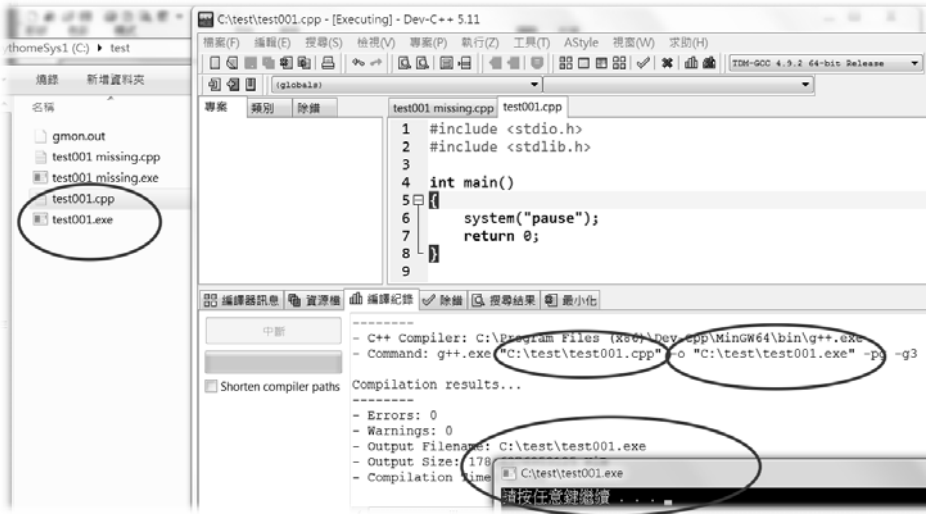
2015/12/28
devcpp 5.11 MingW(gcc 4.8.1, g++ 4.8.1)

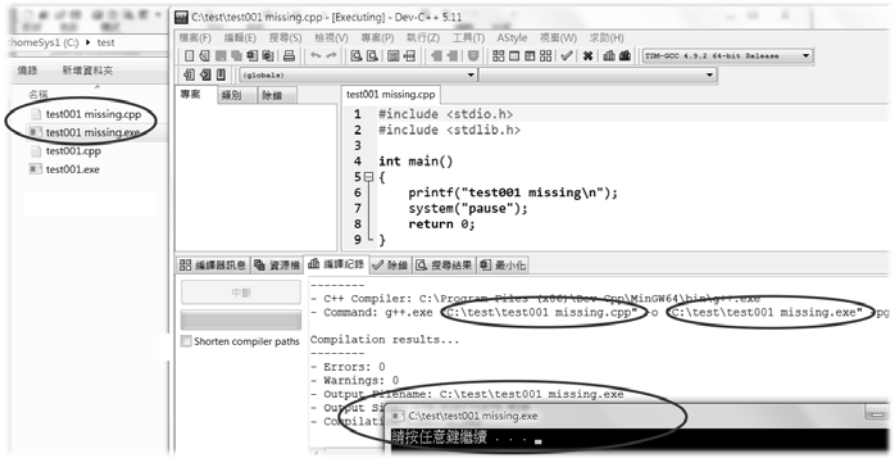
- This is really a nightmare to debug a 300-line program from a novice programmer.
- This is still a major problem for a real software package by an experienced programmer.

Problem #5

- file name with spaces







Disaster!!

Problem #6

- In a project xxx.dev

  yyy.c is compiled by g++ if you select "C++ project" as you create your project

Summary

- I am not demonstrating that Dev C++ is of no use.
- In fact, these problems do not affect my work at all.
  - I seldom commit the first or the third error, at least not when I am sober.
  - I never use the second extended grammar of g++.
- Thus, what I am saying is that Dev C++ is OK for well-trained programmers but clearly not for novice programmers.
- **C** treats you like a "consenting adult" especially the public domain **Dev-C++** and **GNU C/C++**
- Not detecting these problems should be a very serious crime and help cultivating bad habits for a beginning programmer or just destroying his likely shining career.
- Saving money should not be blamed and given in return an inferior compiler.

Other GCC Extensions

- http://tigcc.ticalc.org/doc/gnuexts.html
- Some features that are in ISO C99 but not C89 are also, as extensions, accepted by GCC in C89 mode.
- **Statements and Declarations in Expressions**
- **Locally Declared Labels**
- **Labels as Values**
- **Nested Functions**
- **Constructing Function Calls**
- **Referring to a Type with 'typeof'**
- **Generalized Lvalues**
- **Conditionals with Omitted Operands**
- **Double-Word Integers**
- **Complex Numbers**
- **Hex Floats**
- **Structures With No Members**
- **Arrays of Length Zero**
- **Arrays of Variable Length**
- **Macros with a Variable Number of Arguments**
- **Non-Lvalue Arrays May Have Subscripts**
- **Arithmetic on void and Function Pointers**
- **Non-Constant Initializers**
- **Compound Literals (Cast Constructors)**
- **Designated Initializers**

- **Cast to a Union Type**
- **Case Ranges**
- **Specifying Attributes of Functions**
- **Specifying Attributes of Variables**
- **Specifying Attributes of Types**
- **Attribute Syntax**
- **Prototypes and Old-Style Function Definitions**
- **C++ Style Comments**
- **Dollar Signs in Identifier Names**
- **Escape Character in Constants**
- **Inquiring on Alignment of Types or Variables**
- **Inline Functions**
- **Inline Assembler**
- **Controlling Names Used in Assembler Code**
- **Variables in Specified Registers**
- **Alternate Keywords**
- **Incomplete 'enum' Types**
- **Function Names as Strings**
- **Getting the Return or Frame Address of a Function**
- **Other built-in functions provided by GCC**
- **Slightly Looser Rules for Escaped Newlines**
- **String Literals with Embedded Newlines**
- **Mixed Declarations and Code**
- **Unnamed struct/union Fields within structs/unions**
- **Definite Access of Volatile Objects**
- **History**
- **GNU General Public License**
- **GNU Free Documentation License**
- **Funding Free Software**

Not really a Problem

```
#include <iostream>
#include <string>
#include <math.h>
using namespace std;
int main()
{
    char id[2];
    id[0]='5';
    cout << id[0] << endl;
    cout << (int)id[0] << endl;
    cout << (int)pow(id[0],2) << endl;
    cout << (int)pow(id[0],3) << endl;
    cout << (int)pow(id[0],4) << endl;
    return 0;
}
```

Dev C++

5

53

2809

148876

7890481

VC6

5

53

2809

148877

7890481