



## 高级语言程序设计-II

### Assignment 1

## 1 配置 docker 环境

```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS C:\Users\mix\Documents\Learn\Program\workingspace> docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

PS C:\Users\mix\Documents\Learn\Program\workingspace> |
```

图 1: 运行 docker run hello-world

```
PS C:\Users\mix\Documents\Learn\Program\workingspace> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
scucpphw_test_img   latest          8ed17bf55a57   29 hours ago   1.3GB
gcc                  11.2.0          c2968a627869   10 months ago  1.23GB
hello-world         latest          feb5d9fea6a5   17 months ago  13.3kB
PS C:\Users\mix\Documents\Learn\Program\workingspace> docker ps
CONTAINER ID   IMAGE               COMMAND          CREATED        STATUS        PORTS           NAMES
32514c419ad2   scucpphw_test_img:latest  "/bin/bash"     29 hours ago  Up 5 hours   22/tcp          zen_ramanujan
PS C:\Users\mix\Documents\Learn\Program\workingspace> |
```

图 2: 运行 docker images 和 docker ps

```
Windows PowerShell
PS C:\Users\mix\Documents\Learn\Program\workspace> docker exec -it 32514c419ad2 /bin/bash
root@32514c419ad2:/# cd /ws/code
root@32514c419ad2:/ws/code#
```

图 3: 进入 Linux 内并执行相应操作

## 2 利用 g++ 编译单文件

### 2.1 单步编译与分布编译

```
Windows PowerShell
root@32514c419ad2:/ws/code# g++ test.cpp
root@32514c419ad2:/ws/code# g++ -E test.cpp -o test.i
root@32514c419ad2:/ws/code# g++ -S test.i -o test.s
root@32514c419ad2:/ws/code# g++ -c test.s -o test.o
root@32514c419ad2:/ws/code# g++ test.o -o test
root@32514c419ad2:/ws/code# ./a.out
vector v after call to generate_n() with lambda: 1 1 2 3 5 8 13 21 34
x: 1 y: 1
vector v after 1st call to fillVector(): 1 2 3 4 5 6 7 8 9
vector v after 2nd call to fillVector(): 10 11 12 13 14 15 16 17 18
root@32514c419ad2:/ws/code# ./test
vector v after call to generate_n() with lambda: 1 1 2 3 5 8 13 21 34
x: 1 y: 1
vector v after 1st call to fillVector(): 1 2 3 4 5 6 7 8 9
vector v after 2nd call to fillVector(): 10 11 12 13 14 15 16 17 18
root@32514c419ad2:/ws/code# ls
a.out inefficiency.cpp test test.cpp test.i test.o test.s
root@32514c419ad2:/ws/code#
```

图 4: 单步与分步编译并运行

根据分布编译过程，整个编译过程被划分为 4 个过程：先对程序进行预处理，即得到 `test.i` 的过程；然后编译器将源代码转化为汇编代码，即得到 `test.s` 的过程；接着将汇编代码转化为机器代码，即得到 `test.o` 的过程；最后将库文件与得到的机器代码进行链接，得到最后的可执行文件。

另外，指令 `g++` 的 `-o` 参数表示输出文件名，如果没有，则默认为 `a.out`。

### 2.2 编译过程优化

```
Windows PowerShell
root@32514c419ad2:/ws/code# g++ inefficiency.cpp -o without_o.out
root@32514c419ad2:/ws/code# g++ inefficiency.cpp -O2 -o with_o.out
root@32514c419ad2:/ws/code# time ./without_o.out
result = 100904034

real    0m2.983s
user    0m2.979s
sys     0m0.000s
root@32514c419ad2:/ws/code# time ./with_o.out
result = 100904034

real    0m0.006s
user    0m0.002s
sys     0m0.000s
root@32514c419ad2:/ws/code#
```

图 5: 优化与非优化编译的执行信息

通过两运行时间的对比，可以很明显的观察到，就这个程序而言，使用了优化的编译过程要远快于没有使用优化的编译过程。

## 2.3 其他编译选项

这里选择 `test_class.cpp`, `test_class_size.cpp`, `test_move.cpp`, `test_noexcept.cpp` 分别执行 `-std=c++17`, `-Wall`, `-g` 编译指令并输出调试信息与运行结果。

```
Windows PowerShell
root@32514c419ad2:/ws/code/other_test# ls
test_class.cpp      test_default_parameter.cpp  test_noexcept.cpp  test_raii.cpp
test_class_size.cpp test_move.cpp                test_ptr.cpp
root@32514c419ad2:/ws/code/other_test# g++ -std=c++17 test_class.cpp -o test_class
root@32514c419ad2:/ws/code/other_test# ./test_class
1      #21325302 is created
1      #58320212 is created
5      #21325302      5000      5000
25     #58320212      10000     10000
45     #21325302      5500      10500
60     #58320212      -4000     6000
90     #21325302      27.64     10527.6
90     #58320212      21.78     6021.78
#21325302      Balance: 10527.6
#58320212      Balance: 6021.78
root@32514c419ad2:/ws/code/other_test# g++ -Wall test_class.cpp -o test_class
root@32514c419ad2:/ws/code/other_test# ./test_class
1      #21325302 is created
1      #58320212 is created
5      #21325302      5000      5000
25     #58320212      10000     10000
45     #21325302      5500      10500
60     #58320212      -4000     6000
90     #21325302      27.64     10527.6
90     #58320212      21.78     6021.78
#21325302      Balance: 10527.6
#58320212      Balance: 6021.78
root@32514c419ad2:/ws/code/other_test# g++ -g test_class.cpp -o test_class_debug
root@32514c419ad2:/ws/code/other_test# readelf -S test_class_debug | grep -i debug
[26] .debug_aranges PROGBITS 0000000000000000 00003082
[27] .debug_info PROGBITS 0000000000000000 000030d2
[28] .debug_abbrev PROGBITS 0000000000000000 0000596d
[29] .debug_line PROGBITS 0000000000000000 00005fc2
[30] .debug_str PROGBITS 0000000000000000 0000623b
[31] .debug_rnglists PROGBITS 0000000000000000 000075f2
[32] .debug_line_str PROGBITS 0000000000000000 0000761e
root@32514c419ad2:/ws/code/other_test#
```

图 6: test\_class.cpp

```
Windows PowerShell
root@32514c419ad2:/ws/code/other_test# g++ -std=c++17 test_class_size.cpp -o test_class_size
root@32514c419ad2:/ws/code/other_test# ./test_class_size
Size of Student: 8
Number: 12345678
Level: sophomore
Grade: B
root@32514c419ad2:/ws/code/other_test# g++ -Wall test_class_size.cpp -o test_class_size
root@32514c419ad2:/ws/code/other_test# g++ -g test_class_size.cpp -o test_class_size_debug
root@32514c419ad2:/ws/code/other_test# readelf -S test_class_size_debug | grep -i debug
[26] .debug_aranges PROGBITS 0000000000000000 0000307a
[27] .debug_info PROGBITS 0000000000000000 000030ba
[28] .debug_abbrev PROGBITS 0000000000000000 0000568e
[29] .debug_line PROGBITS 0000000000000000 00005cd7
[30] .debug_str PROGBITS 0000000000000000 00005eac
[31] .debug_rnglists PROGBITS 0000000000000000 000070fe
[32] .debug_line_str PROGBITS 0000000000000000 00007120
root@32514c419ad2:/ws/code/other_test#
```

图 7: test\_class\_size.cpp

```
Windows PowerShell
root@32514c419ad2:/ws/code/other_test# ls
test_class.cpp      test_default_parameter.cpp  test_noexcept.cpp  test_raii.cpp
test_class_size.cpp test_move.cpp                test_ptr.cpp
root@32514c419ad2:/ws/code/other_test# g++ -std=c++17 test_move.cpp -o test_move
root@32514c419ad2:/ws/code/other_test# ./test_move
Process(int&):0
Process(int&&):1
Process(int&&):0
forward(int&&):2
Process(int&):2
forward(int&&):0
Process(int&):0
root@32514c419ad2:/ws/code/other_test# g++ -Wall test_move.cpp -o test_move
test_move.cpp: In function 'void Decltype::test_decltype(T)':
test_move.cpp:174:46: warning: typedef 'iType' locally defined but not used [-Wunused-local-typedefs]
  174 |     typedef typename decltype(obj)::value_type iType;
      |
test_move.cpp: In function 'void Decltype::test()':
test_move.cpp:170:11: warning: 'elem' is used uninitialized [-Wuninitialized]
  170 |     cout << elem << endl;
      |
root@32514c419ad2:/ws/code/other_test# g++ -g test_move.cpp -o tes_move_debug
root@32514c419ad2:/ws/code/other_test# readelf -S tes_move_debug | grep -i debug
[27] .debug_aranges      PROGBITS      0000000000000000 0000308a
[28] .debug_info          PROGBITS      0000000000000000 0000321a
[29] .debug_abbrev         PROGBITS      0000000000000000 000079fa
[30] .debug_line           PROGBITS      0000000000000000 000083ed
[31] .debug_str            PROGBITS      0000000000000000 00008903
[32] .debug_rnglists       PROGBITS      0000000000000000 0000be5f
[33] .debug_line_str       PROGBITS      0000000000000000 0000bf53
root@32514c419ad2:/ws/code/other_test#
```

图 8: test\_move.cpp

```
Windows PowerShell
root@32514c419ad2:/ws/code/other_test# g++ -std=c++17 test_noexcept.cpp -o test_noexcept
test_noexcept.cpp: In function 'void handler()':
test_noexcept.cpp:5:4: error: 'printf_s' was not declared in this scope; did you mean 'printf'?
   5 |     printf_s("in handler\n");
     |     ~~~~~^
test_noexcept.cpp: At global scope:
test_noexcept.cpp:8:15: error: ISO C++17 does not allow dynamic exception specifications
   8 | void f1(void) throw(int) {
     |
test_noexcept.cpp: In function 'void f1()':
test_noexcept.cpp:9:4: error: 'printf_s' was not declared in this scope; did you mean 'printf'?
   9 |     printf_s("About to throw 1\n");
     |     ~~~~~^
test_noexcept.cpp: At global scope:
test_noexcept.cpp:28:6: error: variable or field '__declspec' declared void
  28 | void __declspec(nothrow) f2(void) {
     |
test_noexcept.cpp:28:17: error: 'nothrow' was not declared in this scope
  28 | void __declspec(nothrow) f2(void) {
     |
test_noexcept.cpp: In function 'int main()':
test_noexcept.cpp:45:4: error: 'f2' was not declared in this scope; did you mean 'f5'?
  45 |     f2();
     |     ~^
test_noexcept.cpp:51:7: error: 'printf_s' was not declared in this scope; did you mean 'printf'?
  51 |     printf_s("Caught exception from f4\n");
     |     ~~~~~^
root@32514c419ad2:/ws/code/other_test#
```

图 9: test\_noexcept.cpp

通过输出发现，编译选项 `-Wall` 是在更加严格的条件进行编译，对一些可能会对程序运行出问题的部分，如没有初值就使用这个变量、定义了变量但是并没有使用等进行警告。

值得注意的是 `test_noexcept.cpp` 文件无法正常编译。经过查阅资料，发现文件中的 `printf_s`

与 `__declspec` 等报错的函数和变量均为 Microsoft VC 下的关键字，无法在 Linux 系统上运行。一个直接的结果就是该文件在本机（Windows11）利用 VS 等 IDE 可以正常编译运行，而在 Linux 虚拟机中编译错误。

在 VS2022 中的运行结果如下。

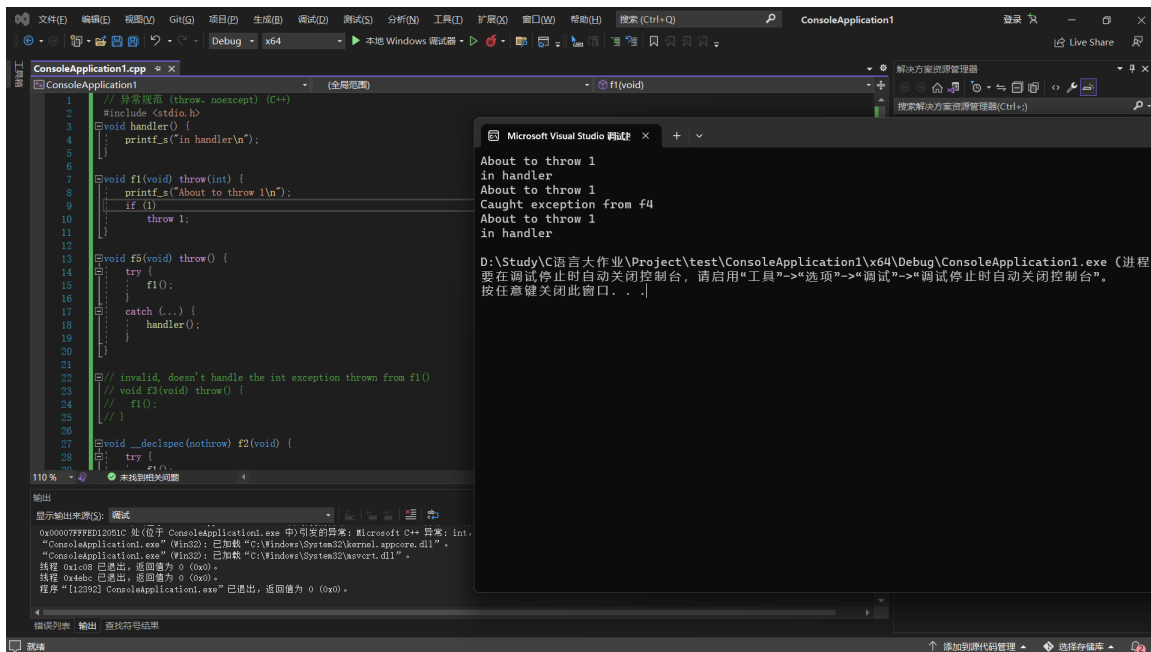


图 10: test\_noexcept.cpp 在 VS2022 中的运行结果

## 3 用 GDB 调试文件

### 3.1 overload.cpp

```
Windows PowerShell
(gdb) start
Temporary breakpoint 4 at 0x40120f: file test_function_overload.cpp, line 14.
Starting program: /ws/code/debugtest/question4/test1
warning: Error disabling address space randomization: Operation not permitted
warning: File "/usr/local/lib64/libstdc++.so.6.0.29-gdb.py" auto-loading has been declined by your 'auto-load safe-path'
set to "$debugdir:$datadir/auto-load".

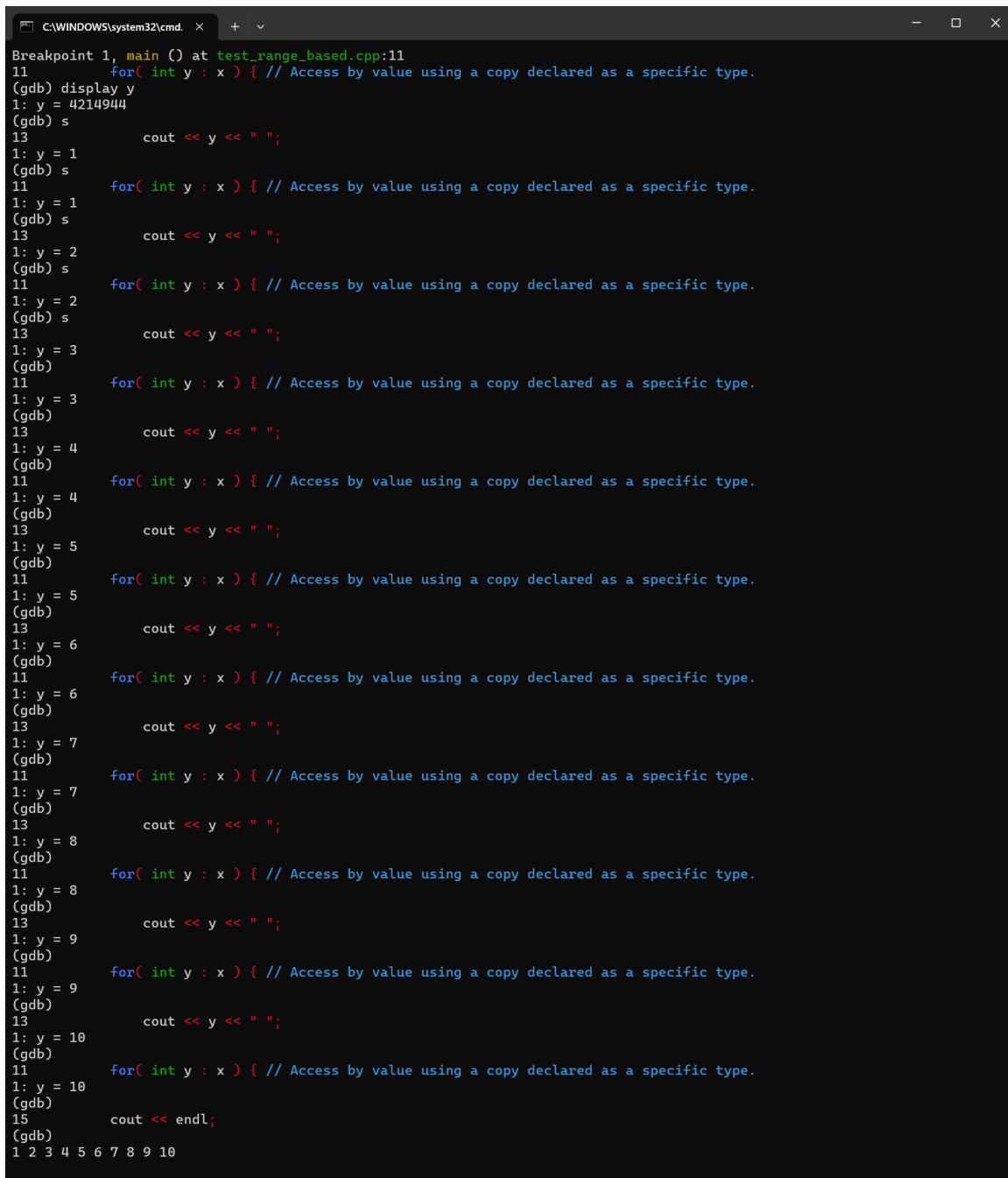
Temporary breakpoint 4, main () at test_function_overload.cpp:14
14      cout << "Enter two integer: ";
(gdb) s
15      cin >> m >> n;
(gdb) s
Enter two integer: 1 2
16      cout << "Their sum of square: " << sumOfSquare(m, n) << endl;
(gdb) s

Breakpoint 1, sumOfSquare (a=1, b=2) at test_function_overload.cpp:5
5      return a * a + b * b;
(gdb) bt
#0 sumOfSquare (a=1, b=2) at test_function_overload.cpp:5
#1 0x0000000000401262 in main () at test_function_overload.cpp:16
(gdb) s
6      }
(gdb) s
Their sum of square: 5
main () at test_function_overload.cpp:19
19      cout << "Enter two real number: ";
(gdb) s
20      cin >> x >> y;
(gdb) s
Enter two real number: 1.2 3.4
21      cout << "Their sum of square: " << sumOfSquare(x, y) << endl;
(gdb) s
sumOfSquare (a=1.2, b=3.3999999999999999) at test_function_overload.cpp:9
9      return a * a + b * b;
(gdb) bt
#0 sumOfSquare (a=1.2, b=3.3999999999999999) at test_function_overload.cpp:9
#1 0x00000000004012d4 in main () at test_function_overload.cpp:21
(gdb) s
10      }
(gdb) n
Their sum of square: 13
main () at test_function_overload.cpp:23
23      return 0;
(gdb) n
24      }
(gdb) n
__libc_start_main (main=0x401206 <main()>, argc=1, argv=0x7ffdc5804048, init=<optimized out>, fini=<optimized out>,
rtld_fini=<optimized out>, stack_end=0x7ffdc5804038) at ../csu/libc-start.c:342
342      ../csu/libc-start.c: No such file or directory.
(gdb) n
[Inferior 1 (process 15512) exited normally]
(gdb)
```

图 11: test\_function\_overload.cpp 的调试

对同一个函数，如果输入不同，那么这几个函数是可以同时存在的，并且根据输入数据的不同，调用不同的函数。

## 3.2 based.cpp



```
Breakpoint 1, main () at test_range_based.cpp:11
11      for(int y : x) { // Access by value using a copy declared as a specific type.
(gdb) display y
1: y = 4214944
(gdb) s
13          cout << y << " ";
1: y = 1
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 1
(gdb) s
13          cout << y << " ";
1: y = 2
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 2
(gdb) s
13          cout << y << " ";
1: y = 3
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 3
(gdb) s
13          cout << y << " ";
1: y = 4
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 4
(gdb) s
13          cout << y << " ";
1: y = 5
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 5
(gdb) s
13          cout << y << " ";
1: y = 6
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 6
(gdb) s
13          cout << y << " ";
1: y = 7
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 7
(gdb) s
13          cout << y << " ";
1: y = 8
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 8
(gdb) s
13          cout << y << " ";
1: y = 9
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 9
(gdb) s
13          cout << y << " ";
1: y = 10
(gdb) s
11      for(int y : x) { // Access by value using a copy declared as a specific type.
1: y = 10
(gdb) s
15          cout << endl;
(gdb)
1 2 3 4 5 6 7 8 9 10
```

图 12: test\_range\_based.cpp 的调试 Part.1

第一个循环 `for( int y : x )`, `y` 的值分别为 1,2,3,4,5,6,7,8,9,10。一开始为未定值 4214944。

```
C:\WINDOWS\system32\cmd. x + v
Breakpoint 2, main () at test_range_based.cpp:17
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
(gdb) display y
2: y = 0
(gdb)
(gdb) s
18          cout << y << " ";
2: y = 1
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 1
(gdb)
18          cout << y << " ";
2: y = 2
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 2
(gdb)
18          cout << y << " ";
2: y = 3
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 3
(gdb)
18          cout << y << " ";
2: y = 4
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 4
(gdb)
18          cout << y << " ";
2: y = 5
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 5
(gdb)
18          cout << y << " ";
2: y = 6
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 6
(gdb)
18          cout << y << " ";
2: y = 7
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 7
(gdb)
18          cout << y << " ";
2: y = 8
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 8
(gdb)
18          cout << y << " ";
2: y = 9
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 9
(gdb)
18          cout << y << " ";
2: y = 10
(gdb)
17      for( auto y : x ) { // Copy of 'x', almost always undesirable
2: y = 10
(gdb)
20      cout << endl;
(gdb)
1 2 3 4 5 6 7 8 9 10
```

图 13: test\_range\_based.cpp 的调试 Part.2

第二个循环 `for( auto y : x )`, `y` 的值同样为 1,2,3,4,5,6,7,8,9,10, 一开始的默认值为 0。



```
C:\WINDOWS\system32\cmd. x + v
Breakpoint 3, main () at test_range_based.cpp:22
22      for( auto &y : x ) { // Type inference by reference.
(gdb) display y
3: y = (int &) @0x7f947be67902: 264275272
(gdb) s
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa860: 1
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa860: 1
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa864: 2
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa864: 2
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa868: 3
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa868: 3
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa86c: 4
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa86c: 4
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa870: 5
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa870: 5
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa874: 6
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa874: 6
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa878: 7
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa878: 7
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa87c: 8
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa87c: 8
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa880: 9
(gdb)
22      for( auto &y : x ) { // Type inference by reference.
3: y = (int &) @0x7fffd661fa880: 9
(gdb)
23          cout << y << " ";
3: y = (int &) @0x7fffd661fa884: 10
(gdb)
25      cout << endl;
(gdb)
1 2 3 4 5 6 7 8 9 10
```

图 14: test\_range\_based.cpp 的调试 Part.3

第三个循环 `for( auto &y : x)`, `y` 的值有前缀 `(int &)@0x7ffdc2569d0`, 后面接 `:1`, 冒号前的十六进制依次加 4, 其后的数依次加 1 直到到 10。初始值为 `(int &) @0x7f947be67902: 264275272`。

这里猜测后面的数表示 `y` 的值; 由于有 `&` 引用符号, 故猜测这里的地址就是数组 `x` 每个元素的地址。

下面将源代码中数组 `x` 的每个元素的地址输出, 发现与上述 `y` 的地址一样, 从而确定了这里的 `y` 是对数组 `x` 中每个元素的一个引用。

```
C:\WINDOWS\system32\cmd. x + v
root@32514c419ad2:/ws/code/debugtest/question4# g++ test_range_based.cpp -o test2
root@32514c419ad2:/ws/code/debugtest/question4# ./test2
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
0x7ffee460e5a0 0x7ffee460e5a4 0x7ffee460e5a8 0x7ffee460e5ac 0x7ffee460e5b0 0x7ffee460e5b4 0x7ffee460e5b8 0x7ffee460e5bc
0x7ffee460e5c0 0x7ffee460e5c4
end of integer array test

0.14159 1.14159 2.14159 3.14159 4.14159 5.14159 6.14159 7.14159 8.14159 9.14159
end of vector test
```

图 15: 数组 x 元素的地址

这里的引用也表明如果在循环内更改  $y$  的值，那么对应数组元素的值也会同时改变。

```
C:\WINDOWS\system32\cmd. x + v
Breakpoint 4, main () at test_range_based.cpp:27
27         for( const auto &y : x ) { // Type inference by const reference.
(gdb) display y
4: y = (const int &) <error reading variable>
(gdb) s
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa860: 1
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa860: 1
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa864: 2
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa864: 2
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa868: 3
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa868: 3
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa86c: 4
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa86c: 4
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa870: 5
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa870: 5
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa874: 6
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa874: 6
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa878: 7
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa878: 7
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa87c: 8
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa87c: 8
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa880: 9
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa880: 9
(gdb)
28             cout << y << " ";
4: y = (const int &) @0x7ffd661fa884: 10
(gdb)
27         for( const auto &y : x ) { // Type inference by const reference.
4: y = (const int &) @0x7ffd661fa884: 10
(gdb)
30             cout << endl;
(gdb)
1 2 3 4 5 6 7 8 9 10
31             cout << "end of integer array test" << endl;
```

图 16: test\_range\_based.cpp 的调试 Part.4

第四个循环 `for( const auto &y : x )` 中, `y` 在 `gdb` 中的调试结果与第三个循环一致, 但是数据类型变为 `const int &`。

至于 `const` 的前缀, 表明 `y` 是一个常量, 与第三个循环的区别在于这里不能修改 `y` 的值, 否则会出现编译错误。同时使用 `const` 修饰会适当加快程序运行的速度。

而 `j` 的值输出为:

---

```
1 0.14159 1.14159 2.14159 3.14159 4.14159 5.14159 6.14159 7.14159 8.14159 9.14159
```

---

通过观察可以得出  $j = \pi - i + 3 (i = 0, 1, 2, \dots, 9)$ 。

通过对源代码内的数据类型和调试过程中的数据类型对比，可以看出 `auto` 关键字是一种自适应的变量类型，根据它所赋的初值决定它的数据类型。代码中的 `auto` 就被处理成 `int` 使用。

### 3.3 const.cpp

```
Temporary breakpoint 1, main () at test_const.cpp:24
24      cout << &p << endl;
(gdb) s
(gdb) s
0x402008
26      cout << &c.NUM << endl;
(gdb) s
0x402004
28      cout << C::NUM1 << endl;
(gdb) s
3
30      int a=5, b=0;
(gdb) s
31      cout<<MAX(++a, b)<<endl;
(gdb) s
7
32      cout<<MAX(++a, b+10)<<endl;
(gdb) s
10
33      a=5,b=0;
(gdb) s
34      cout<<Max(++a,b)<<endl;
(gdb) s
Max<int> (a=@0x7ffdb1aee038: 6, b=@0x7ffdb1aee034: 0) at test_const.cpp:20
20      return (a>b ? a:b);
(gdb) n
21      }
(gdb) n
6
main () at test_const.cpp:35
35      }
(gdb) n
__libc_start_main (main=0x401186 <main()>, argc=1, argv=0x7ffdb1aee138, init=<optimized out>, fini=<optimized out>, rtd_fini=<optimized out>,
stack_end=0x7ffdb1aee128) at ../csu/libc-start.c:342
342      ../csu/libc-start.c: No such file or directory.
(gdb) n
[Inferior 1 (process 16431) exited normally]
(gdb) |
```

图 17: test\_const.cpp 的调试

通过对代码的修改和编译,当对 `enum` 取地址时编译错误,对 `#define p "hello"` 和 `static const int NUM=3` 取地址都能够成功。

对 `p` 取得的地址为 `0x402008`; 对 `NUM` 取得的地址为 `0x402004`。

通过与同学的交流讨论，发现并不是所有宏定义 `#define` 定义的值都有地址，如果它是一个整型，那么它在编译的时候会直接报错；而对于一个字符串，那么它可以正确的编译并且正确的输出。但有两点疑问通过查阅资料并没有解决：一是为什么整形的宏定义没有地址，而字符串有；二是这个地址具体表示的是哪个地址？

## 4 参考资料

<https://www.zhihu.com/question/31965546/answer/2806978357>

[https://blog.csdn.net/nb\\_zsy/article/details/122739253](https://blog.csdn.net/nb_zsy/article/details/122739253)