C++面向对象程序设计 作业报告1

报告人：邓雍　学号：2022141220184

一.体会小结

往日掌握的主要是对C++语言的语法基础认知，以及对于算法的学习。今天在作业的调试与理解中更深入地掌握了linux环境配置的一些相关知识，以及关于docker镜像的一些应用。gdb调试指令的初步掌握也是本次作业中学到的内容。

二.Docker环境配置

在配置指南相对指引下，通过docker网络的重定向完成了对相关系统资源的下载。实行了windows下对linux系统的镜像使用。

下面给出相关实验步骤截图配置

1.对hello-world的程序运行

```
C:\Users\Administrator>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/


C:\Users\Administrator>
```

此步正常运行反映了docker软件的正常使用

2.docker images检查docker配置文件的齐全

```
C:\Users\Administrator>docker images
REPOSITORY       TAG       IMAGE ID       CREATED             SIZE
<none>           <none>    9a031df766fc   About an hour ago   1.28GB
scucpphw_image   latest    f40a000a6d93   11 days ago         1.3GB
gcc              11.2.0    c2968a627869   10 months ago       1.23GB
hello-world      latest    feb5d9fea6a5   17 months ago       13.3kB
```

反映了Docker文件的正常下载

3.docker ps检查映射情况

```
PS C:\Users\Administrator\Desktop> docker ps
CONTAINER ID   IMAGE                   COMMAND       CREATED         STATUS         PORTS     NAMES
e09030d46bd9   scucpphw_image:latest   "/bin/bash"   4 seconds ago   Up 3 seconds   22/tcp    nifty_tu
```

4.docker映射的检查

```
PS C:\Users\Administrator\Desktop> docker exec -it e09030d46bd9 /bin/bash
root@e09030d46bd9:/# cd /ws/code/
root@e09030d46bd9:/ws/code#
```

三.g++编译相关问题

1.g++单步编译与分步编译问题

```
root@e09030d46bd9:/ws/Assignment_1_test/question3# g++ test.cpp
root@e09030d46bd9:/ws/Assignment_1_test/question3# ls
a.out  inefficency.cpp  other_test  test.cpp
```

图1为简单直接的单步编译展示，在没有写明参数的情况下，会编译为默认的程序名a.out

```
root@e09030d46bd9:/ws/Assignment_1_test/question3# g++ -E test.cpp -o test.i
root@e09030d46bd9:/ws/Assignment_1_test/question3# ls
a.out  inefficency.cpp  other_test  test.cpp  test.i
root@e09030d46bd9:/ws/Assignment_1_test/question3#  g++ -S test.i -o test.s
root@e09030d46bd9:/ws/Assignment_1_test/question3# ls
a.out  inefficency.cpp  other_test  test.cpp  test.i  test.s
root@e09030d46bd9:/ws/Assignment_1_test/question3# g++ -c test.s -o test.o
root@e09030d46bd9:/ws/Assignment_1_test/question3# ls
a.out  inefficency.cpp  other_test  test.cpp  test.i  test.o  test.s
root@e09030d46bd9:/ws/Assignment_1_test/question3# g++ test.o -o test
root@e09030d46bd9:/ws/Assignment_1_test/question3# ls
a.out  inefficency.cpp  other_test  test  test.cpp  test.i  test.o  test.s
root@e09030d46bd9:/ws/Assignment_1_test/question3#
```

图二为分步编译运行情况的展示，分别写明了编译，汇编，链接这三步操作与操作后的文件夹情况展示

```
root@e09030d46bd9:/ws/Assignment_1_test/question3# ./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@e09030d46bd9:/ws/Assignment_1_test/question3# ./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
```

图三为对编译内容所得出的可执行文件的检验，可以发现单步编译与分步编译编译所得的效果是相同的。

2.优化编译对运行效率的影响

```
root@e09030d46bd9:/ws/Assignment_1_test/question3# g++ inefficency.cpp -o without_o.out
root@e09030d46bd9:/ws/Assignment_1_test/question3# g++ inefficency.cpp -O2 -o with_o.out
root@e09030d46bd9:/ws/Assignment_1_test/question3# time ./with_o.out
result = 100904034

real    0m0.011s
user    0m0.004s
sys     0m0.000s
root@e09030d46bd9:/ws/Assignment_1_test/question3# time ./without_o.out
result = 100904034

real    0m3.361s
user    0m3.345s
sys     0m0.010s
```

可以观察到在O2加速的情况下，运行时间极大地缩短了，体现了优化编译对时间的高效节省

3.其他cpp文件的不同编译信息测试

```
root@e09030d46bd9:/ws/Assignment_1_test/question3# cd other_test/
root@e09030d46bd9:/ws/Assignment_1_test/question3/other_test# g++ -std=c++17 test_class.cpp -o test1.out
root@e09030d46bd9:/ws/Assignment_1_test/question3/other_test# ls
test1.out       test_class_size.cpp         test_move.cpp      test_ptr.cpp
test_class.cpp  test_default_parameter.cpp  test_noexcept.cpp  test_raii.cpp
root@e09030d46bd9:/ws/Assignment_1_test/question3/other_test# ./test1.out
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
```

-std=C++17的编译指令可以使cpp文件按照C++17的版本进行编译，版本影响程序运行的效率，也与新STL使用有所关联

```
root@e09030d46bd9:/ws/Assignment_1_test/question3/other_test# g++ -Wall test_move.cpp -o test2.out
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@e09030d46bd9:/ws/Assignment_1_test/question3/other_test# ./test2.out
Process(int&):0
Process(int&&):1
Process(int&&):0
forward(int&&):2
Process(int&):2
forward(int&&):0
Process(int&):0
```

-Wall命令可以显示警告信息，方便调试一些可能由手误导致的错误。例如变量类型不对应等问题

```
root@e09030d46bd9:/ws/Assignment_1_test/question3/other_test# g++ -g test_ptr.cpp -o test3.out
root@e09030d46bd9:/ws/Assignment_1_test/question3/other_test# ls
a.out      test2.out  test_class.cpp       test_default_parameter.cpp  test_noexcept.cpp  test_raii.cpp
test1.out  test3.out  test_class_size.cpp  test_move.cpp               test_ptr.cpp
root@e09030d46bd9:/ws/Assignment_1_test/question3/other_test# ./test3.out
weak1 is expired
5
```

-g命令可以显示调试信息，配合gdb使用可以更高效方便的调试代码

四.gdb调试代码的应用

1.函数栈帧与层级查看

```
root@e09030d46bd9:/ws/Assignment_1_test/question4# g++ -g test_function_overload.cpp -o test1
root@e09030d46bd9:/ws/Assignment_1_test/question4# gdb test1
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from test1...
(gdb)
```

加入-g命令编译并gdb可执行文件进入调试界面

```
(gdb) b sumOfSquare
Breakpoint 1 at 0x4011c0: sumOfSquare. (2 locations)
(gdb) r
Starting program: /ws/Assignment_1_test/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".
To enable execution of this file add
        add-auto-load-safe-path /usr/local/lib64/libstdc++.so.6.0.29-gdb.py
line to your configuration file "/root/.gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/root/.gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"
Enter two integer: bt

Breakpoint 1, sumOfSquare (a=0, b=4198608) at test_function_overload.cpp:5
5               return a * a + b * b;
(gdb) continue
Continuing.
Their sum of square: 1763354880

Breakpoint 1, sumOfSquare (a=2.0747120802178963e-317, b=0) at test_function_overload.cpp:9
9               return a * a + b * b;
(gdb) continue
Continuing.
Enter two real number: Their sum of square: 0
[Inferior 1 (process 1664) exited normally]
(gdb)
```

在函数中间设置断点，并在断点触发时使用bt指令查询函数栈帧与层级

2.追踪特定变量值

```
root@457ba46ceb43:/ws/Assignment_1_test/question4# g++ -g test_range_based.cpp -o test2
root@457ba46ceb43:/ws/Assignment_1_test/question4# gdb test2
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from test2...
```

加入-g命令编译并gdb可执行文件进入调试界面

```
(gdb) b 12
Breakpoint 1 at 0x4012b3: file test_range_based.cpp, line 13.
(gdb) b 18
Breakpoint 2 at 0x40131d: file test_range_based.cpp, line 18.
(gdb) b 23
Breakpoint 3 at 0x401386: file test_range_based.cpp, line 23.
(gdb) b 28
Breakpoint 4 at 0x4013f2: file test_range_based.cpp, line 28.
(gdb) r
Starting program: /ws/Assignment_1_test/question4/test2
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".
To enable execution of this file add
        add-auto-load-safe-path /usr/local/lib64/libstdc++.so.6.0.29-gdb.py
line to your configuration file "/root/.gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/root/.gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
(gdb) display y
1: y = 1
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 2
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 3
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 4
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 5
```

```
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 6
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 7
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 8
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 9
(gdb) c
Continuing.

Breakpoint 1, main () at test_range_based.cpp:13
13              cout << y << " ";
1: y = 10
(gdb) c
Continuing.
1 2 3 4 5 6 7 8 9 10

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
(gdb) display y
2: y = 1
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 2
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 3
```

```
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 4
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 5
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 6
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 7
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 8
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 9
(gdb) c
Continuing.

Breakpoint 2, main () at test_range_based.cpp:18
18              cout << y << " ";
2: y = 10
(gdb) c
Continuing.
1 2 3 4 5 6 7 8 9 10

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
(gdb) display y
3: y = (int &) @0x7ffd65c87d60: 1
```

```
(gdb) c
Continuing.

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d64: 2
(gdb) c
Continuing.

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d68: 3
(gdb) c
Continuing.

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d6c: 4
(gdb) c
Continuing.

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d70: 5
(gdb) c
Continuing.

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d74: 6
(gdb) c
Continuing.

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d78: 7
(gdb) c
Continuing.

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d7c: 8
(gdb) c
Continuing.

Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d80: 9
(gdb) c
Continuing.
```

```
Breakpoint 3, main () at test_range_based.cpp:23
23              cout << y << " ";
3: y = (int &) @0x7ffd65c87d84: 10
(gdb) c
Continuing.
1 2 3 4 5 6 7 8 9 10

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
(gdb) display y
4: y = (const int &) @0x7ffd65c87d60: 1
(gdb) c
Continuing.

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d64: 2
(gdb) c
Continuing.

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d68: 3
(gdb) c
Continuing.

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d6c: 4
(gdb) c
Continuing.

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d70: 5
(gdb) c
Continuing.

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d74: 6
(gdb) c
Continuing.

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d78: 7
(gdb) c
Continuing.
```

```
Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d7c: 8
(gdb) c
Continuing.

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d80: 9
(gdb) c
Continuing.

Breakpoint 4, main () at test_range_based.cpp:28
28              cout << y << " ";
4: y = (const int &) @0x7ffd65c87d84: 10
(gdb) c
Continuing.
1 2 3 4 5 6 7 8 9 10
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
[Inferior 1 (process 1552) exited normally]
```

在循环中设置断点，并在进入循环时使用display指令显示某特定变量具体值，利用C指令继续运行函数至下一次触发断点。

从函数本身输出可以看出对普通数组取地址输出与对vector取地址输出是不同的。最后打印的j即是对vector中元素的引用，对应vector中相应的准确值

3.代码修改与地址改错

```
class C {
public:

    static const int NUM = 3;
    enum con {
        NUM1 = 3
    };
};

#define MAX(a,b) ((a) > (b) ? (a) : (b))
template<typename T>
inline int Max(const T& a, const T& b){
    return (a>b ? a:b);
}
const int C::NUM;
int main() {
    cout << p << endl;
    C c;
    cout << &C::NUM << endl;

    cout << C::NUM1 << endl;

    int a=5, b=0;
    cout<<Max(++a, b)<<endl;
    cout<<Max(++a, b+10)<<endl;
    a=5,b=0;
    cout<<Max(++a,b)<<endl;
}
```

观察到代码本身有相当多的错误

主要分为三点

1.static const int是整个结构体类型对应一个地址，不存在该结构体变量每个对应一个值，故不存c.NUM的地址，应改写为C::NUM

2.NUM1是结构体中的no-const变量，应该是每个c有单独NUM1地址

3.define的MAX函数是将整个(a)表达式拿进去重写，导致++a会本身出现两次，导致了重复的累加，故应该改写为普通的Max

```
root@457ba46ceb43:/ws/Assignment_1_test/question4# g++ -g test_const.cpp -o test3
root@457ba46ceb43:/ws/Assignment_1_test/question4# ./test3
hello
0x402004
3
6
10
6
```

附上修改后的函数运行结果

总结：const有地址,enum与#define没有地址，上图中已打印const地址