1 Stack, Static And Heap

请按照要求，产生可执行文件，并给出每一个步骤的截图。以下要求使用 Linux 或者 Windows 的 GCC 工具。

1.1 测试一

以下包括代码和操作。

1.1.1 `main1.c` 文件

```c
#include<stdio.h>
int num = 3;
void inc_num()
{
   num++;
   printf("addr of num in inc(): %d\n", &num);
   printf("increase num to be: %d\n", num);
}
void dec_num()
{
   num--;
   printf("addr of num in dec(): %d\n", &num);
   printf("decrease num to be: %d\n", num);
}
int main()
{
   int a; short b; char c;
   printf("Initially num = %d\n", num);
   printf("addr of num in main(): %d\n", &num);
   inc_num();
   inc_num();
   dec_num();
   inc_num();
   printf("addr of a: %p\n", &a);
   printf("addr of b: %p\n", &b);
   printf("addr of c: %p\n", &c);
   return 0;
```

```
28 }
```

1.1.2 步骤

1. 根据以上代码产生可执行文件，并运行得到输出结果。
2. 在 `inc_num()`、`dec_num()` 和 `main()` 函数中获取 `num` 的地址，分别得到什么？有什么规律？
3. 在 `inc_num()` 和 `dec_num()` 函数除了输出之外，产生了何种影响？
4. 程序中的变量（`num`、`a`、`b`、`c`），它们的地址分别是什么？有何规律？

1.2 测试二

以下包括代码和操作。

1.2.1 `main2.c` 文件

```c
#include<stdio.h>
void call()
{
    short b = 0;
    static int call_time = 0;
    printf("in call(), addr of b: %p\n", &b);
    printf("in call(), addr of call_time: %p\n", &call_time);
    call_time++;
    printf("I am the %d-th call.\n", call_time);
}

void f()
{
    call();
}

int main()
{
```

```
19    int a;
20    call();
21    f();
22    call();
23    printf("addr of a: %p\n", &a);
24    return 0;
25 }
```

1.2.2 步骤

1. 根据以上代码产生可执行文件，并运行得到输出结果。
2. `f()` 和 `call()` 分别被调用多少次?
3. 按开始被调用的先后顺序给出写出各个被调用的函数（`main()` 函数可以不写）。
4. 按调用结束的先后顺序给出写出各个被调用的函数（`main()` 函数可以不写）。
5. 在 `call()` 的每一次被调用中，`b` 和 `call_time` 的地址分别是什么? 有何规律? 背后的原理是什么?
6. 第 19 行的变量 `a` 的地址是什么? 结合 `b` 和 `call_time` 的地址，这里有何规律? 背后的原理是什么?

1.3 测试三

以下包括代码和操作。

1.3.1 `main3.c` 文件

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  double *multiplyByTwo (double *ptr_to_input)
5  {
6     printf("In multiplyByTwo():\n");
```

```
7     printf("ptr_to_input: %p\n", ptr_to_input);
8     printf("addr of ptr_to_input: %p\n", &ptr_to_input);
9
10    double *ptr_to_twice;
11
12    ptr_to_twice = malloc(sizeof(double));
13    printf("ptr_to_twice: %p\n", ptr_to_twice);
14    printf("addr of ptr_to_twice: %p\n", &ptr_to_twice);
15
16    *ptr_to_twice = (*ptr_to_input) * 2.0;
17
18    printf("Below outside multiplyByTwo():\n");
19    return ptr_to_twice;
20 }
21
22 int main()
23 {
24    printf("In main():\n");
25    int *ptr_to_age;
26
27    ptr_to_age = malloc(sizeof(int));
28    printf("ptr_to_age: %p\n", ptr_to_age);
29    printf("addr of ptr_to_age: %p\n", &ptr_to_age);
30
31    *ptr_to_age = 30;
32    double *ptr_to_salary;
33
34    //
35    ptr_to_salary = malloc(sizeof(double));
36    printf("ptr_to_salary: %p\n", ptr_to_salary);
37    printf("addr of ptr_to_salary: %p\n", &ptr_to_salary);
38
39    //
40    *ptr_to_salary = 12345.25;
41
42    double *ptr_to_my_extra_work_time;//
43
44    ptr_to_my_extra_work_time = malloc(3 * sizeof(double));
45    printf("ptr_to_my_extra_work_time: %p\n",
```

```
        ptr_to_my_extra_work_time);
46  printf("addr of ptr_to_my_extra_work_time: %p\n",
        &ptr_to_my_extra_work_time);
47
48  ptr_to_my_extra_work_time[0] = 1.0;
49  ptr_to_my_extra_work_time[1] = 2.25;
50  ptr_to_my_extra_work_time[2] = 3.5;
51
52  double *ptr_to_twice_salary;
53
54  ptr_to_twice_salary = multiplyByTwo(ptr_to_salary);
55
56  printf("double your salary is %.2lf\n",
        *ptr_to_twice_salary);
57
58  free(ptr_to_age);
59  free(ptr_to_salary);
60  free(ptr_to_my_extra_work_time);
61  free(ptr_to_twice_salary);
62
63  printf("about to leave main()\n");
64
65  return 0;
66 }
```

1. 根据以上代码产生可执行文件，并运行得到输出结果。
2. 考虑 `ptr_to_input`、`ptr_to_twice`、`ptr_to_age`、`ptr_to_salary` 和 `ptr_to_my_extra_work_time`,请写出它们的地址以及它们的值。
3. 上述变量的地址以及它们的值，存在什么规律？请简述原因。
4. 上述代码在 heap 空间获取空间最多是在什么时候？该时候一共获取了多少空间？
5. 第 58~61 行的顺序可否打乱？请简述原因。

1.4 实验报告写作要求

1. 步骤详细；
2. 表述简明；
3. 图文并茂；
4. 逻辑流畅。