中山大學
SUN YAT-SEN UNIVERSITY

程序设计II实验

电子与信息工程学院（微电子学院）

庞志勇 高级实验师 助教：焦涵 、桂海田 博士研究生

PPT大纲

- 上次课实验总结
- 本次课的实验介绍

上次课实验总结

- “老师您好！我想问问我们上周的程设二实验是要把PPT上面的三个实验复现吗？毕竟PPT上都已经有源代码了”
- 编程没有所谓的“标准答案”，不能只会老师PPT上的答案。
- 第一次课我们要求每道题目除了自己程序，还要网上书上找至少3个程序进行比较，做到“多读多写”！
- 除了老师PPT上内容，同学们要自己做的。“博学”。
- 编程本来就没有什么标准答案一说，每个人的编程习惯不同，编程的思维不同，最后写出的代码肯定都是不一样的。
- 编程代码不会写，这个，我想说，没有人一开始就是会的，都是自己不断学习，找网上技术大牛的代码功能实现，模仿，学习别人解决问题的思路，举一反三，自己再利用这种思路来解决自己所面对的编程难题。作者：Ada链接：https://www.zhihu.com/question/427693503/answer/2237282864来源：知乎著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

学、问、思、辨、行

上次课实验总结

编译器问题：

- 同样的程序，在不同编译器是否通过，编译结果可能不同
- Dev C++和VC6.0不同
- 可能支持的C++标准不同
- Printf在VS2022上要特殊处理。。。

上次课实验总结

题目描述: 输入一个正整数n，计算1! + 2! + 3! + ... + n! (0 < n < 10)
输入描述: 一个正整数n (0 < n < 10)
输出描述: 1！ ~n!的阶乘的和
样例输入: 5
样例输出: 153

问题分析与描述（算法）

第一种方法：利用嵌套循环。外部循环用来求和，内部循环用来计算阶乘。需要注意的是，本道题需要一个变量来存放每个数的阶乘的值，每次将阶乘的值计入总和之后，需要将其重置为1，以便不影响最后的结果。

第一种方法：利用嵌套循环

- 数据类型
- 标识符
- 常量
- 变量
- 运算符和表达式
- 语句
- 输入输出函数
- 顺序选择循环结构程序

```
#include <iostream>
using namespace std;

int main(){
        int sum = 0,factorial = 1,n;//sum阶乘和变量，factorial阶乘变量

        cout<<"Please enter an integer n:"<<endl;
        cin>>n;

        for(int i = 1;i <= n;i++){     //外循环计算1!到n!的阶乘和
                for(int j = 1;j <= i;j++){  //内循环计算n!的阶乘
                        factorial *= j;  //注意对复合赋值运算理解好
                }
                sum += factorial;
                factorial = 1;
        }

        cout<<"The sum of n factorial is "<<sum<<endl;
        return 0;
}
```

第二种方法：这种方法确实是最简单的一种方法，是基于第一种方法而改进的。在第一种方法中，我们利用嵌套循环来分别进行求阶乘和求和，并用一个变量存放每个数阶乘的值，每次循环最后还要将这个变量重置为1。其实没有必要将这个变量重置为1，仔细思考就会发现：

1的阶乘是1：1！=1

2的阶乘是1*2，也就是1的阶乘乘2：2！=1！*2

那3的阶乘就是1*2*3，也就是2的阶乘乘3：3！=2！*3

以此类推，i的阶乘就是i-1的阶乘乘i，那i+1的阶乘就是i的阶乘乘i+1，所以我们在循环中，只要用上一个数的阶乘乘下一个数，就是下一个数的阶乘，实现此思想的代码如下：

```
#include<iostream>
using namespace std;
int main(){
        int n,jie = 1,sum = 0;

        cin>>n;

        for(int i = 1 ;i <= n;i++){
                jie *= i;
                sum += jie;
        }

        cout<<sum;

        return 0;
}
```

第三种方法：我们利用递归函数来计算阶乘，只用一个for循环即可。

- 数据类型
- 标识符
- 常量
- 变量
- 运算符和表达式
- 语句
- 输入输出函数
- 顺序选择循环结构程序
- 函数

```
#include <iostream>
using namespace std;
int factorial(int n){                    //递归求n！的阶乘
        if(n ==1)
                return 1;
        else
                return n* factorial(n-1);
}
int main(){
        int n, sum = 0;

        cin>>n;

        for(int i = 1;i <= n;i++){
                sum += factorial(i);
        }

        cout<<sum;
        return 0;
}
```

第四种方法：我们利用函数来计算阶乘。尽量避免使用递归函数。我们要会C和C++，尤其是对于我们电信专业的学生一定要学会C语言，因为后续课程很多要用到C。C语言在嵌入式设计中，效率更佳。

```
#include<stdio.h>
unsigned long Fact(unsigned int n);
int main(void)
{
    int m;
        do{
        printf("Input m(m>0):");
        scanf("%d", &m);
    }while(m<0);//增加对输入数据的限制
    printf("%d!=%lu\n", m, Fact(m));
    return 0;
}
unsigned long Fact(unsigned int n)
{//不能在函数里检查输入数据，因为n已经被定义为无符号整型变量
    unsigned int i;
    unsigned long result = 1;
    for(i=2;i<=n;i++)
    {
        result = result * i;
    }
    return result;
}
```

上次课实验总结

- 通过上面阶乘求和程序的四种不同实现算法，我们会发现每道编程题，可以有很多不同的算法
- 我们在第一次课时强调每道编程题，除了自己写的代码，还有书上网上找至少3个代码分析优缺点，做到“多读多写”，这样才能真正学会程序设计
- 算法复杂度不同，运算时间，存储空间。。。后续《数据结构和算法》课程会重点讲解

本次课要做哪些实验？

中山大學 SUN YAT-SEN UNIVERSITY

第4章 面向过程编程实验

- 实验一、VC6使用与cout输出程序设计
- 实验二、数据类型、常量、变量、表达式
- 实验三、输入输出流
- 实验四、选择结构程序设计
- 实验五 循环结构程序设计
- 实验六 控制结构综合实验
- 实验七 函数实验
- 实验八 作用域、生存期及函数实验
- 实验九 数组实验
- 实验十 指针实验
- 实验十一 结构体（记录）实验

课程设计I

第5章 面向对象编程实验

- 实验一 类与对象
- 实验二 函数重载与运算符重载
- 实验三 继承与派生
- 实验四 多态性与虚函数
- 实验五 模板与STL
- 实验六 流类库与文件操作
- 实验七 异常处理

课程设计II

- 实验内容根据理论课内容进行调整
- 每次课实验内容可以根据自己学习情况动态调整
- 课程设计可以自选其他内容

实验六 作用域、生存期及函数实验

一、实验目的

1. 了解变量的作用域、生存期和可见性。
2. 理解变量和函数的声明、定义、调用的区别
3. 递归函数
4. 函数重载、函数模板

实验六 作用域、生存期及函数实验

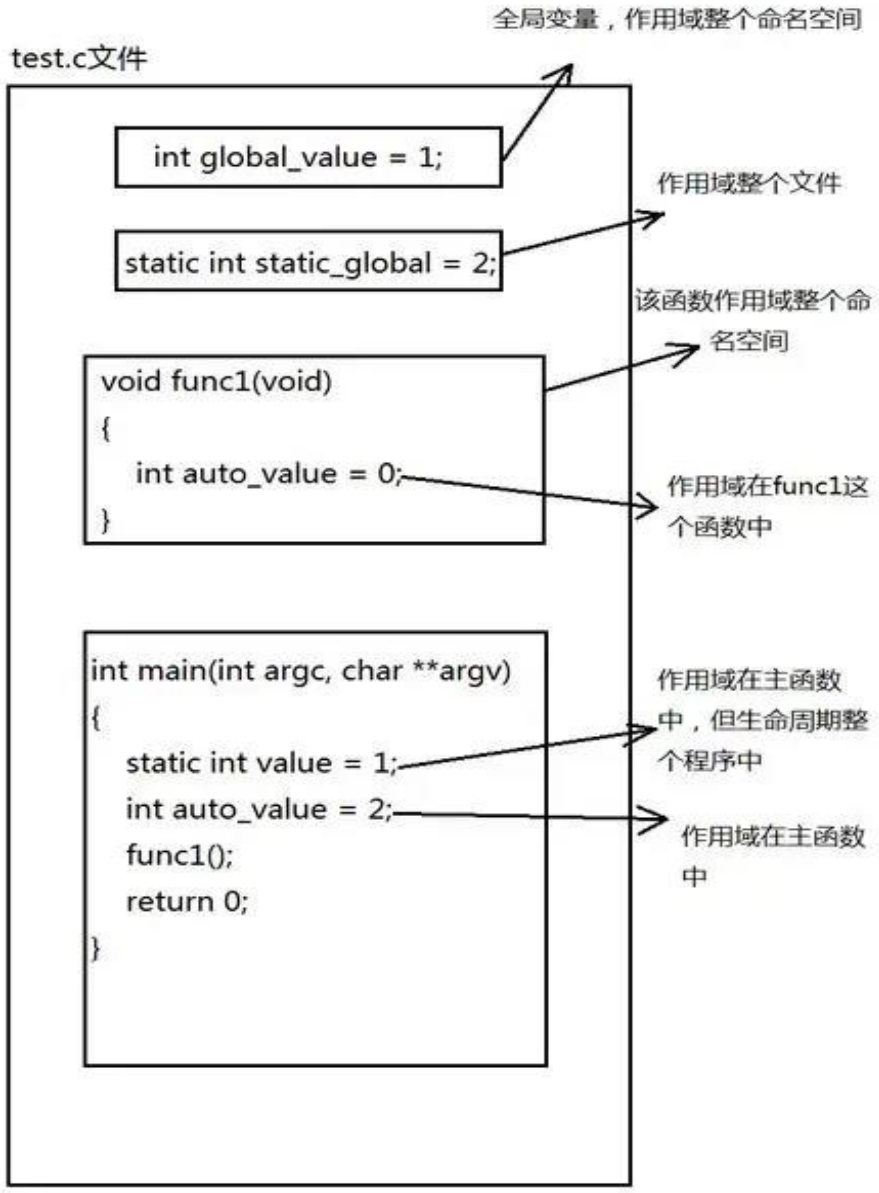
二、实验原理

- 任何一种编程中，作用域是程序中定义的变量所存在的区域，超过该区域变量就不能被访问。
- 作用域是指标识符在程序中可见的部分，可以看作是标识符的一个有效范围。按其作用域范围可分为全局作用域和局部作用域。
- 全局作用域表现为标识符从定义处开始可以在整个程序或其他程序中进行引用
- 局部作用域则只能在局部范围内引用。
- C++作用域分以下几类：类作用域、局部作用域、命名空间作用域、全局作用域。
- C++函数名称具有全局作用域，如果变量和常数的名称也在所有函数和命名空间之外声明也具有全局作用域。
- 在块内{ }声明的变量和常量具有局部作用域，它们在块外不可见。
- 函数的参数与函数最外层块中声明的局部变量具有相同的作用域。
- 如果嵌套块内部和外部包含具有相同名称的本地声明标识符，嵌套块内部标识符具有名称优先级，覆盖外部标识符，即就近原则。

实验六 作用域、生存期及函数实验

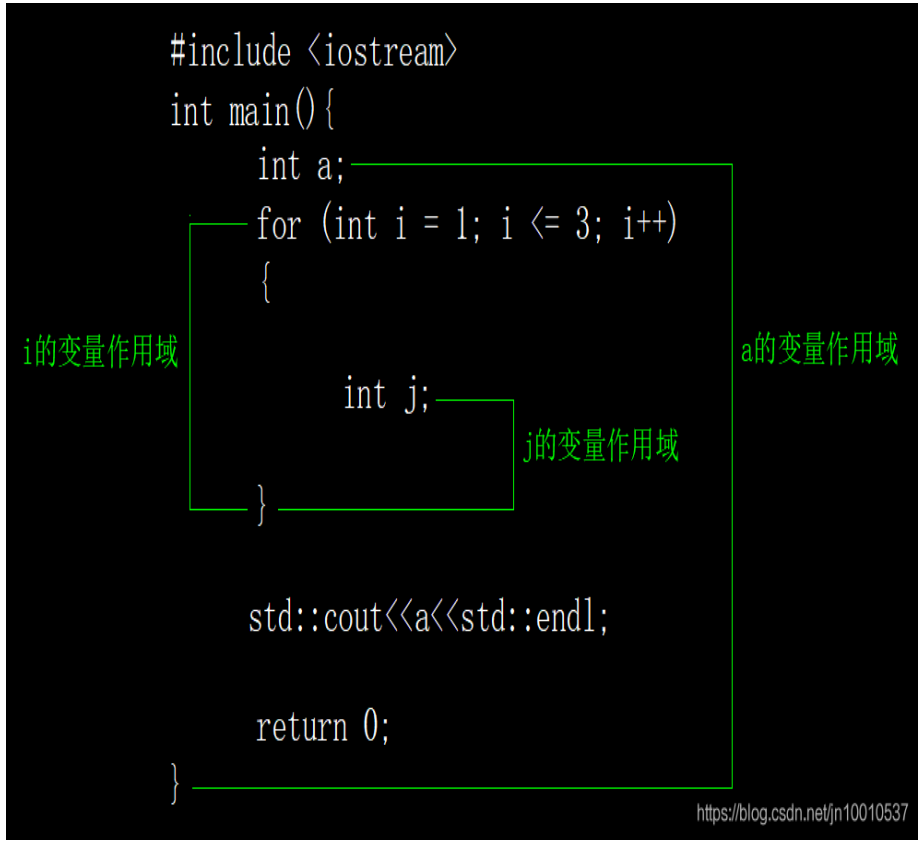
二、实验原理：变量作用域（一图胜千言）

```
int a1;
char a2;

int main()
{

}

void Block1( int   a1,
             char& b2 )
{
    int c1;
    int d2;

}

void Block2()
{
    int a1;
    int b2;

    while (...)
    {           // Block3
        int c1;
        int b2;
    }

}
```



实验六 作用域、生存期及函数实验

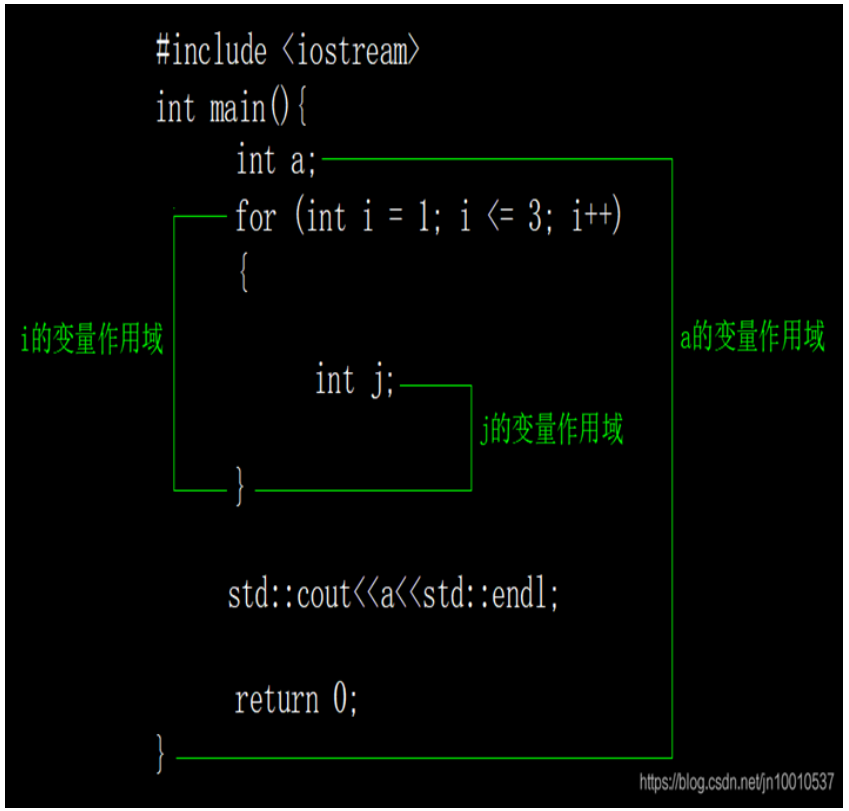
二、实验原理：变量作用域（一图胜千言）




```
int x; //global x

void f()
{
      int x; // local x hides global x
      x=1; // assign to local x
     {
          int x; // hides first local x
          x=2; // assign to second local x
      }
      x=3; // assign to first local x
}
int* p = &x;  // take address of global x
```

变量a属于全局变量，变量i，j属于局部变量

注意：不同编译器编译结果可能不一致




```
#include<stdio.h>
#include<iostream>

using namespace std;

int main()
{
        for(int i=1;i<4;i++)
        {
          int j=8;
        }
        i=8;// VC6.0 中for循环int i定义作用域到这里了
        cout<<"i="<<i<<endl;
        printf("i=%d\n",i);
        //int i;   //error C2086: 'i' : redefinition
        //j=11;  //error C2065: 'j' : undeclared identifier
}
```

实验六 作用域、生存期及函数实验

三、实验内容1

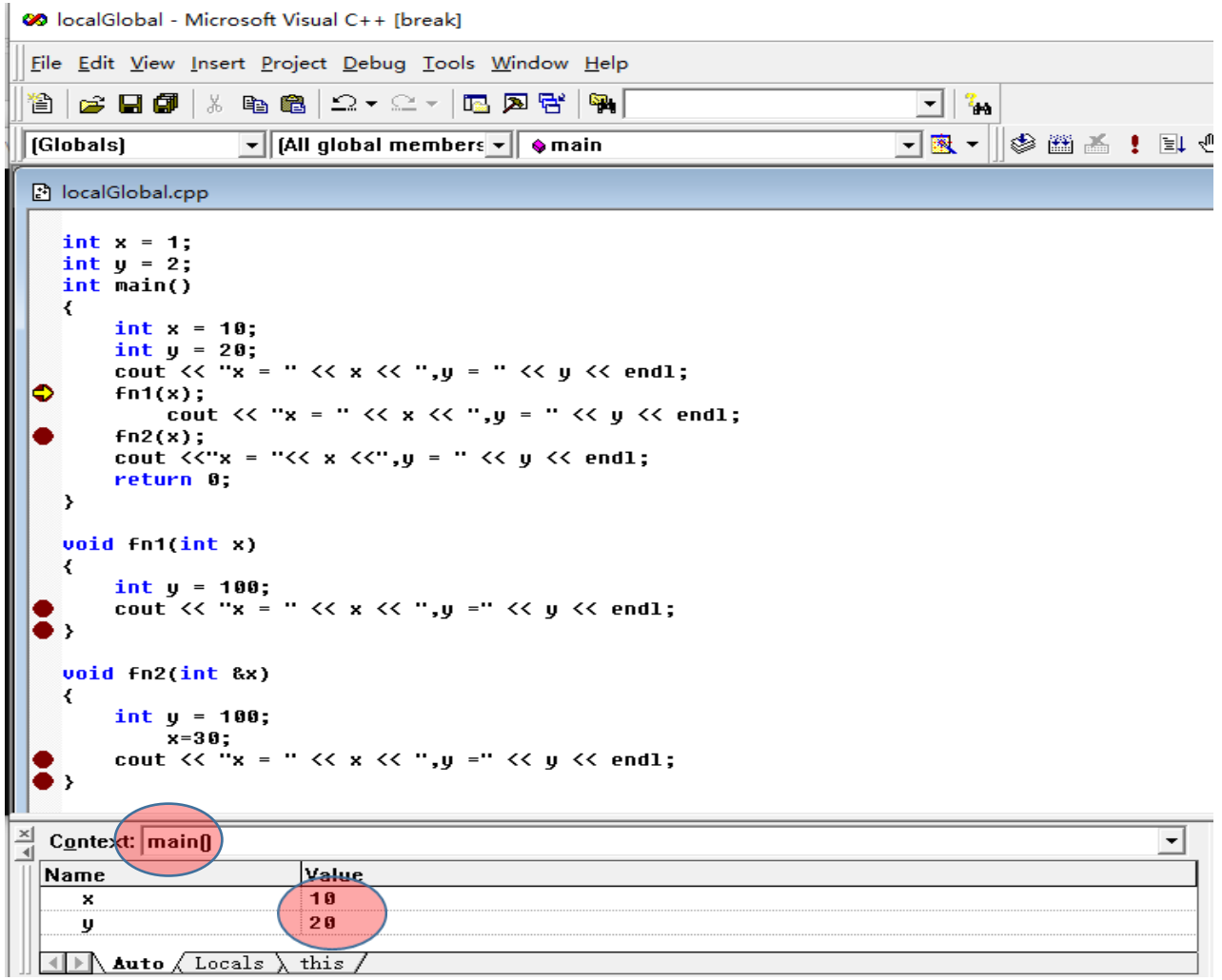
1．注释出下面程序中的全局变量和局部变量，分析程序运行结果,说明结果的原因，运用VC6.0断点调试方法显示执行结果，说明该程序设计主要缺点。

```
#include <iostream>
using namespace std;

void fn1();     //函数声明，函数重载
void fn1(int x);
void fn2(int &x);

int x = 1;
int y = 2;
int main()
{
        int x = 10;
        int y = 20;
        cout << "x = " << x << ",y = " << y << endl;
        fn1(x);
        cout << "x = " << x << ",y = " << y << endl;
        fn2(x);
        cout <<"x = "<< x <<",y = " << y << endl;
        return 0;
}

void fn1(int x)    //函数定义
{
        int y = 100;
        cout << "x = " << x << ",y =" << y << endl;
}
void fn2(int &x)
{
        int y = 100;
         x=30;
        cout << "x = " << x << ",y =" << y << endl;
}
```

实验步骤：

设置好断点，主要断点设置的位置，一般设置在要观察的变量前后，然后每次按F5进行执行到断点操作，观察执行结果，和自己预期结果比较，加深理解。

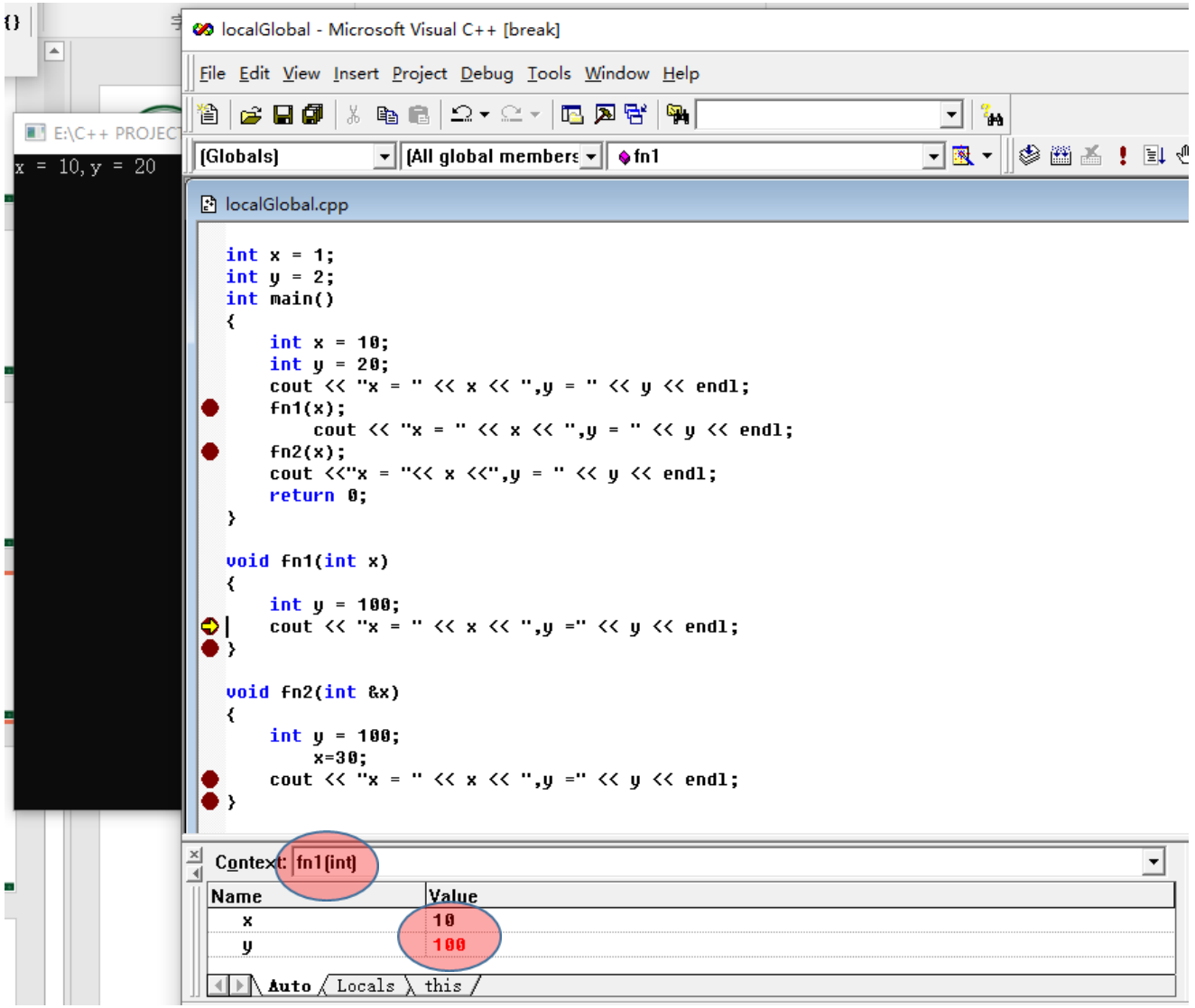
第一次按F5，执行到第一个断点停下，注意观察此时x=10，y=20的值对应的是main()主函数。屏蔽了全局变量x=1,y=2。同时也注意观察此时控制台窗口输出。



实验步骤：

第二次按F5，执行到哪个断点？

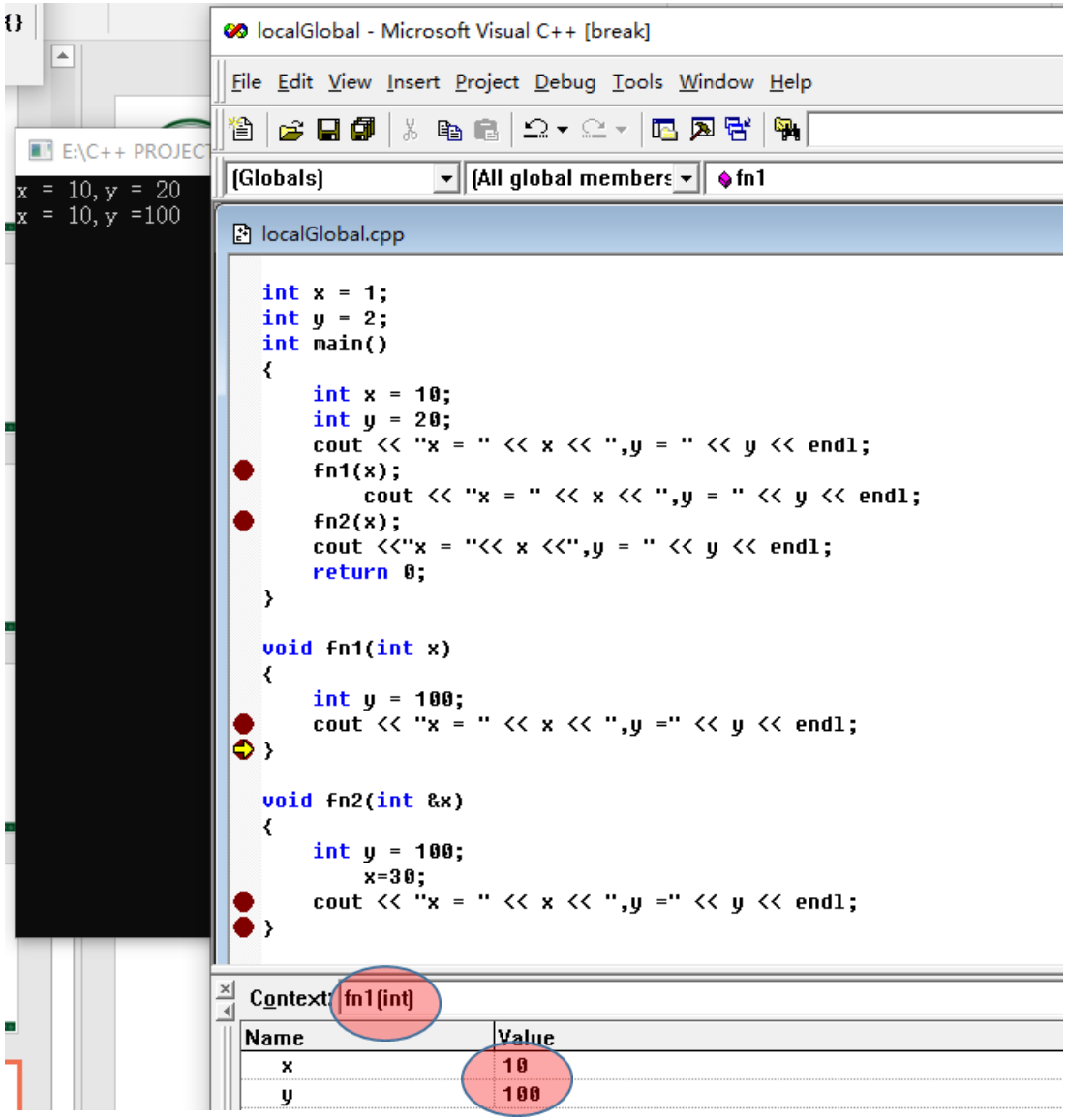
注意此时执行的函数是fn1，对应的变量值也应该是fn1函数内的局部变量x，y的值。Fn1只对y付了新值y=100，x=10还是main函数值保持不变。同时也注意观察此时控制台窗口输出。



实验步骤：

第三次按F5，执行到哪个断点？

注意此时执行的函数是fn1，对应的变量值也应该是fn1函数内的局部变量x，y的值。屏蔽了全局变量x，y和main函数变量x，y。同时也注意观察此时控制台窗口输出。

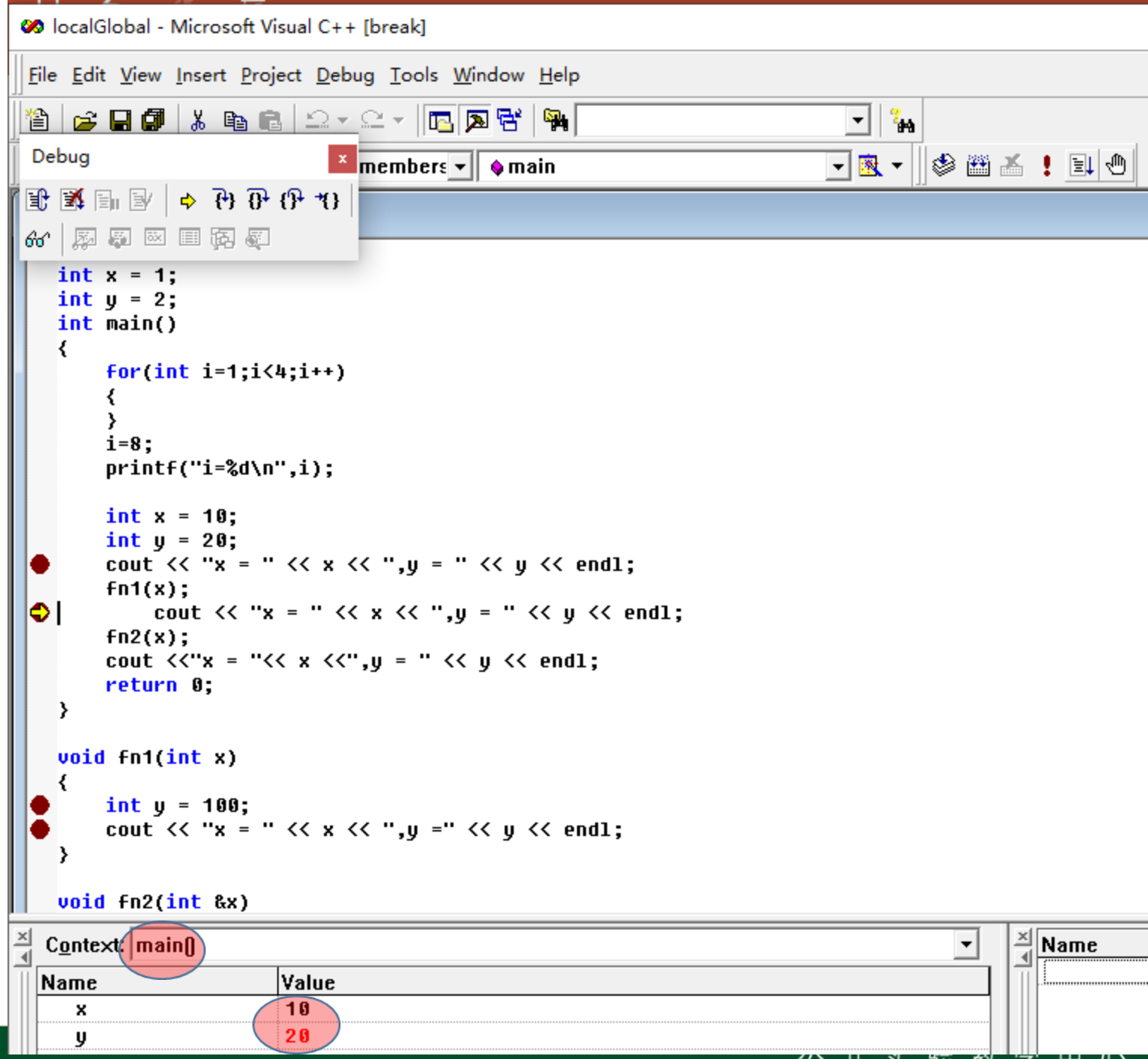


中山大學 SUN YAT-SEN UNIVERSITY

实验步骤：

第四次按F5，执行到哪个断点？

注意此时执行的函数是main，对应的变量值也应该是main函数内的局部变量x=10，y=20的值。同时也注意观察此时控制台窗口输出。



分析程序缺点

实验步骤：

- 显而易见，变量命名没有选用有意义的名字，x，y很多重复。
- 很容易引起变量赋值和实际想法相背离。错误很难发现。
- 使用全局变量，不知道全局变量什么时候被修改了。有经验的程序员会尽可能避免使用全局变量。

实验六 作用域、生存期及函数实验

三、实验内容

2．编写C++程序，实现读入一篇英文文件，正确打印输出每一行的字数、对应的行号和总字数。

流类各自的功能分别为：
- ✓ istream：常用于接收从键盘输入的数据；
- ✓ ostream：常用于将数据输出到屏幕上；
- ✓ ifstream：用于读取文件中的数据；
- ✓ ofstream：用于向文件中写入数据；
- ✓ iostream：继承自 istream 和 ostream 类，因为该类的功能兼两者于一身，既能用于输入，也能用于输出；
- ✓ fstream：兼 ifstream 和 ofstream 类功能于一身，既能读取文件中的数据，又能向文件中写入数据。

```
void readTxt(string file)
{
    ifstream infile;
    infile.open(file.data());   //将文件流对象与文件连接起来
    assert(infile.is_open());   //若失败,则输出错误消息,并终止
程序运行

    string s;
    while(getline(infile,s))
    {
        cout<<s<<endl;
    }
    infile.close();             //关闭文件输入流
}
```

实验六 作用域、生存期及函数实验

三、实验内容2

1．编写C++程序，实现读入一篇英文文件，正确打印输出每一行的字数、对应的行号和总字数。

◆ getline()此函数可读取整行，包括前导和嵌入的空格，并将其存储在字符串string对象中。

1） istream& getline (istream &is , string &str , char delim);

2） istream& getline (istream &is , string &str);

delim 终结符，遇到该字符停止读取操作，不写的话默认为回车。

如：第一种为可以定义的间隔符，第二种为回车结束，注意：1、当getline与whlie一起使用的时候，即while(getline(cin,str))是一个死循环，只有当按Ctrl+z或键入EOF才会结束。2、当使用getline的时候，如果前面用cin输入东西，则必须输入cin.get()吃掉回车符

◆ 当我们需要知道字符串长度时，可以调用 string 类提供的 length() 函数，例如：

```
string s = "http://c.biancheng.net";
int len = s.length();
cout<<len<<endl;
```

实验六 作用域、生存期及函数实验

三、实验内容2

2．分析下面程序代码错误原因，进行运行调试，使得程序能够读入一篇英文文件，正确打印输出每一行的字数和对应的行号。要求报告中给出单步调试函数截图和步骤说明。

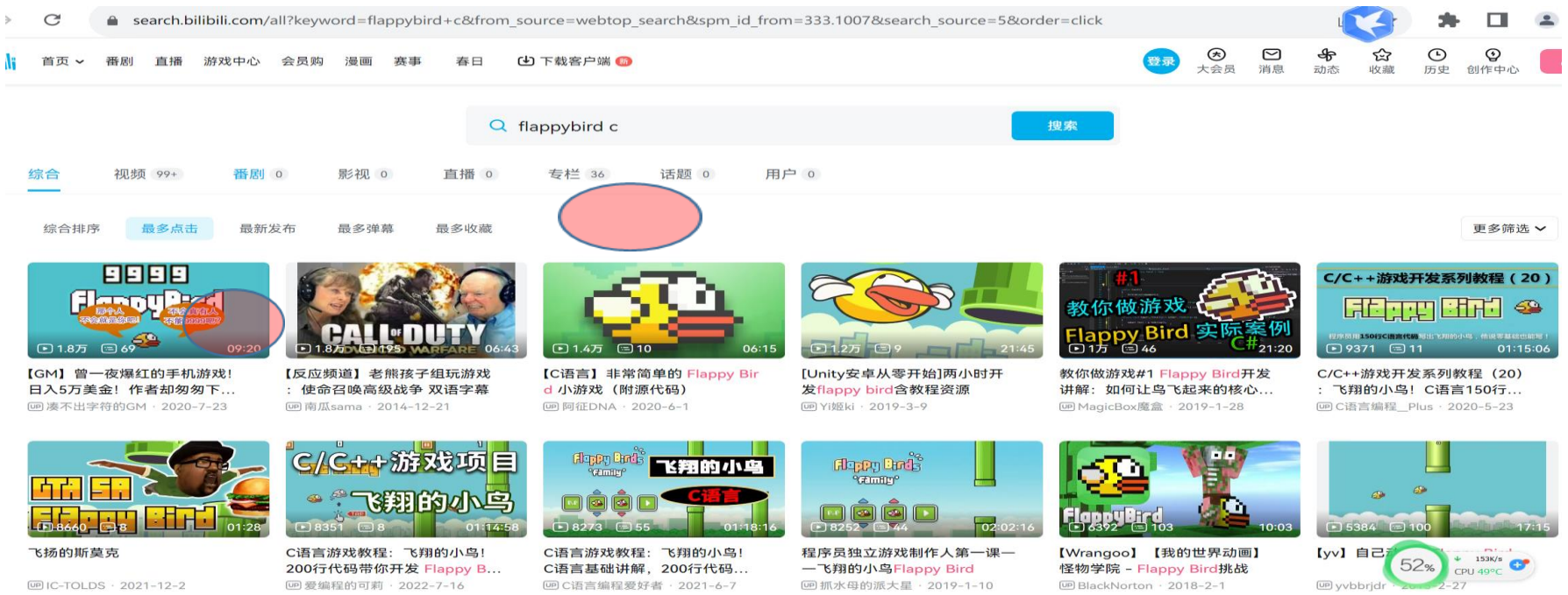
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

void LetterCount(string line, int& count);
//Nunber of letters in line is returned in count
void LineCount(istream& f1le, int& count);
//Nunber of lines in f1le is returned in count

int main()
{
        ifstream inFile;
        inFile.open("test.dat");
        int count = 0;
        LineCount(inFile, count);
        cout << "Number of lines: " << count << endl;
        return 0;
}
```

```
void LetterCount(string line,int& count)
{
        count = line.length();
        cout << " has " << count << " characters；" << endl;
}

void LineCount(istream& file, int& count)
{
        string line;
        while (file)
        {
                getline(file, line);
                count++;
                cout << "Line " << count;
                LetterCount(line,count);
        }
}
```

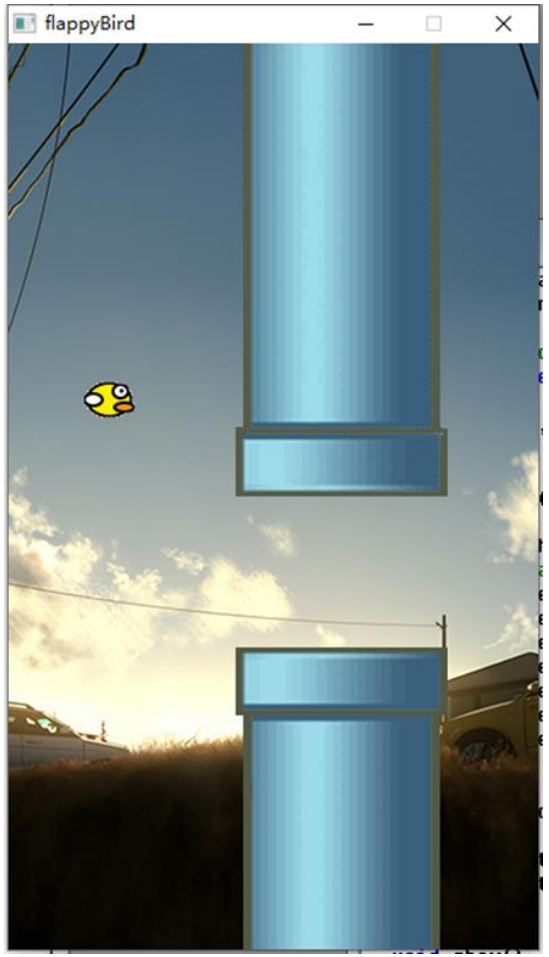
实验六 作用域、生存期及函数实验

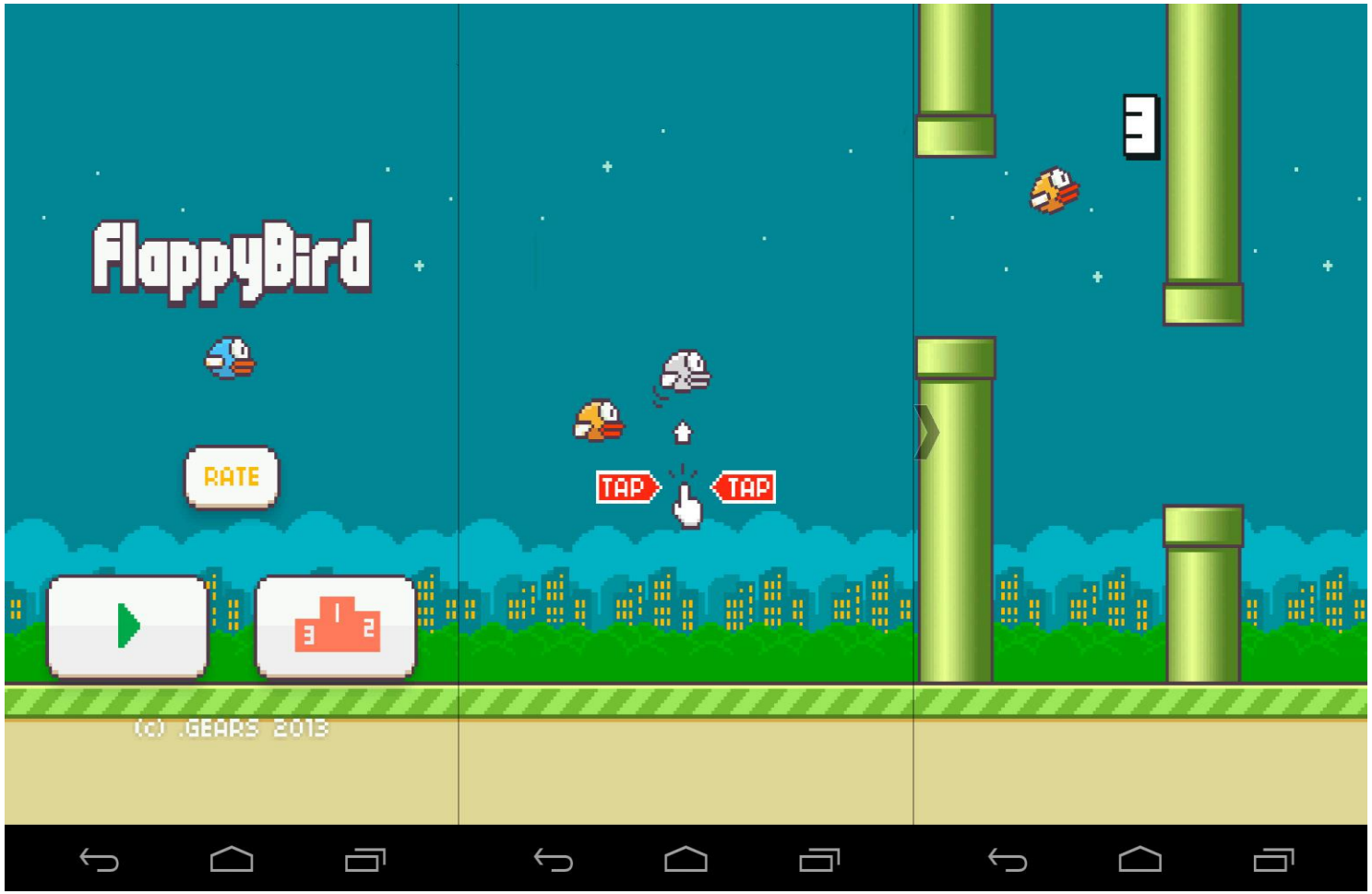
三、实验内容3 Flappybird游戏EasyX

https://search.bilibili.com/all?vt=15317116&keyword=flappybird+c%E8%AF%AD%E8%A8%80&from_source=webtop_search&spm_id_from=333.1007&search_source=5&order=click



FlappyBird





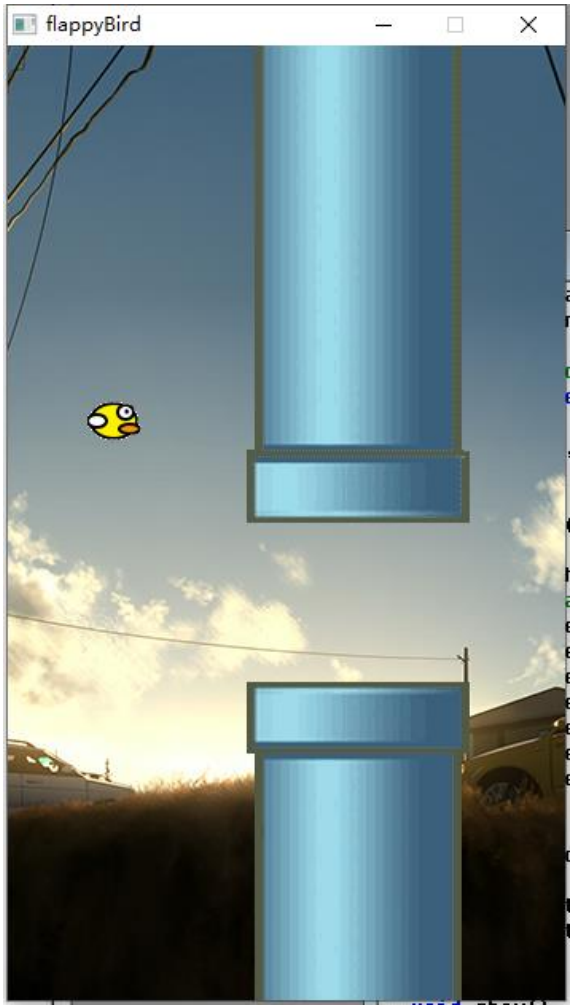
利用函数进行自顶向下设计

- 没有函数，我们所有的代码都要写在主函数中，这样看着杂乱也不利于更新和添加功能。不适合进行更大规模程序开发。

- 我们将给出一个模板，体现自顶向下设计

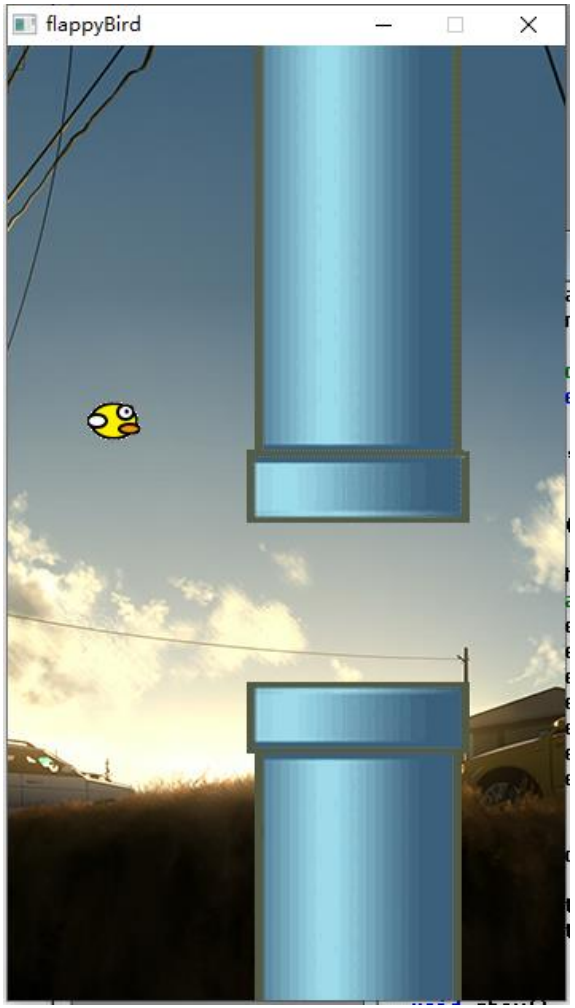
```
int main() {
    startup();//初始化
    while (true) {//游戏循环开始
        show();//显示画面
        updateWithoutInput();//与用户输入无关的更新
        updateWithInput();//与用户输入有关的更新
    }
    return 0;
}
```

1.图片导入（输入）利用loadimage函数（体会函数的力量）




```
void startup()
{
	initgraph(350, 600);
//	loadimage(&img_bk, "background.jpg");//如果图片在工程目录下，可以不加目径
	loadimage(&img_bk, "C:\\C Project\\flappyBird\\background.jpg");
	loadimage(&img_bd1, "C:\\C Project\\flappyBird\\bird1.jpg");
	loadimage(&img_bd2, "C:\\C Project\\flappyBird\\bird2.jpg");
	loadimage(&img_bar_up1, "C:\\C Project\\flappyBird\\bar_up1.gif");
	loadimage(&img_bar_up2, "C:\\C Project\\flappyBird\\bar_up2.gif");
	loadimage(&img_bar_down1, "C:\\C Project\\flappyBird\\bar_down1.gif");
	loadimage(&img_bar_down2, "C:\\C Project\\flappyBird\\bar_down2.gif");
	bird_x = 50;
	bird_y = 200;
	BeginBatchDraw();

	mciSendString("open D:\\background.mp3 alias bkmusic", NULL, 0, NULL);开背景音乐
	mciSendString("play bkmusic repeat", NULL, 0, NULL);  // 循环播放
}
```

2.图片显示（输出）利用putimage函数（体会函数的力量）




```
void show()
{
        putimage(0, 0, &img_bk);    // 显示背景
        putimage(150, -300, &img_bar_up1,NOTSRCERASE); // 显示上一半的障碍
物
        putimage(150, -300, &img_bar_up2,SRCINVERT);
        putimage(150, 400, &img_bar_down1,NOTSRCERASE); // 显示下一半的障
碍物
        putimage(150, 400, &img_bar_down2,SRCINVERT);
        putimage(bird_x, bird_y, &img_bd1,NOTSRCERASE); // 显示小鸟
        putimage(bird_x, bird_y, &img_bd2,SRCINVERT);
        FlushBatchDraw();
        Sleep(50);
}
```

3.小鸟自由下落函数

```
void updateWithoutInput()  //小鸟下落
{
        if (bird_y<500)
                bird_y = bird_y+3;
}
```

4.按键控制函数，空格键小鸟上跳

```
void updateWithInput()
{
        char input;
        if(kbhit()) // 判断是否有输入
        {
                input = getch();
                if (input == ' ' && bird_y>20)
                {
                        bird_y = bird_y - 60;
                        mciSendString("close jpmusic", NULL, 0, NULL); // 先把前面一次的音乐关闭
                        mciSendString("open D:\\Jump.mp3 alias jpmusic", NULL, 0, NULL); // 打开跳动音乐
                mciSendString("play jpmusic", NULL, 0, NULL); // 仅播放一次
                }
        }
}
```

5.游戏画面结束函数

```
void gameover()
{
        EndBatchDraw();
        getch();
        closegraph();
}
```

6.main主函数

```
int main()
{
        startup();  // 数据初始化
        while (1)  //  游戏循环执行
        {
                show();  // 显示画面
                updateWithoutInput();  // 与用户输入无关的更新
                updateWithInput();    // 与用户输入有关的更新
        }
        gameover();     // 游戏结束、后续处理
        return 0;
}
```

讨论分析上述用函数进行程序设计的好处

- 一个大规模c语言程序分为很多子函数（模块）主要是为了程序便于调试，以及程序更趋于结构化，增强了程序的可读性和可移植性，减少代码冗余。
- C、C++提供标准函数库给用户，这些函数可方便的调用，可以减少用户不必要的开发工作量。

主动学习创新能力FlappyBird程序改进

- Flappybird的程序其实和前面我们做的“别碰方块”程序很类似，学习要多思考，做到“触类旁通”
- 如何改进Flappybird程序?

三、实验内容4把飞机游戏代码按照实验内容3的方法，改成函数版本进行实现

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main()
{
        int i,j;
        int x = 5;
        int y = 10;
        char input;
        int isFire = 0;

        int ny = 5; // 一个靶子，放在第一行，ny列上
        int isKilled = 0;

        while (1)
        {
                system("cls");   // 清屏函数

                if (!isKilled)  // 输出靶子
                {
                        for (j=0;j<ny;j++)
                                printf(" ");
                        printf("+\n");
                }

                if (isFire==0) // 输出飞机上面的空行
                {
                        for(i=0;i<x;i++)
                                printf("\n");
                }
                else   // 输出飞机上面的激光竖线
                {
                        for(i=0;i<x;i++)
                        {
                                for (j=0;j<y;j++)
                                        printf("
```

的正中间，距最左边2个坐标

子

输入来移动，不必输入回车

```
                }
                return 0;
        }
                if (y+2==ny)  // +2是因为激光在飞机
                                isKilled = 1; // 击中靶

                isFire = 0;
        }

        // 下面输出一个复杂的飞机图案
        for (j=0;j<y;j++)
                printf(" ");
        printf("  *\n");
        for (j=0;j<y;j++)
                printf(" ");
        printf("*****\n");
        for (j=0;j<y;j++)
                printf(" ");
        printf(" * * \n");


        if(kbhit())  // 判断是否有输入
        {
                input = getch();  // 根据用户的不同

                if (input == 'a')
                        y--;  // 位置左移
                if (input == 'd')
                        y++;  // 位置右移
                if (input == 'w')
                        x--;  // 位置上移
                if (input == 's')
                        x++;  // 位置下移
                if (input == ' ')
                        isFire = 1;
        }
```

学习的四重境界

知学、好学、会学、乐学

祝大家实验顺利！