CS1001A.10
习题课01

SA21011004
郭雨轩

常见编译错误01

```c
#include <stdio.h>
int main(void) {
    printf("%d", i); // error line
    return 0;
}
```

```
test.c:3:18: error: use of undeclared identifier 'i'
```

错误原因：使用了未声明的标识符

注：在该代码段中，变量i应该先声明在引用

常见编译错误02

```c
#include <stdio.h>
int main(void) {
    prinf("Hello world!"); // error line
    return 0;
}
```

```
test.c:3:5: error: implicit declaration of function 'prinf'
is invalid in C99
```

错误原因：隐式函数声明在C99规范中不可用

注：在该代码段中，printf函数被错误输入为了prinf，编译器并不认识这个符号，被认为成隐式函数声明

常见编译错误03

```c
#include <stdio.h>
int main(void) {
    int a = 0;
    int *b;
    b = a; // error line
    return 0;
}
```

```
test.c:5:7: warning: incompatible integer to pointer
conversion assigning to 'int *' from 'int';
```

错误原因：不匹配的整形到指针类型的转换

注：在该代码段中，a是整形变量，b是指针型变量，两者不能直接赋值，直接赋值会导致后续的对于b的内存访问操作访问到非法的地址

常见编译错误04

```c
#include <stdio.h>
int main(void) {
    int a = 0 // error line
    return 0;
}
```

```
test.c:3:14: error: expected ';' at end of declaration
```

错误原因：行尾未加';'

注：在一些旧版本的编译器报错中，可能再下一行的位置提示上一行结尾期望有一个';'

常见编译错误05

```c
#include <stdio.h>
int main(void) {
    for (int i = 0, i < 10; ++i) ; // error line
    return 0;
}
```

```
test.c:3:23: error: expected ';' in 'for' statement
specifier
```

错误原因：for循环头中错误使用','分割语句，应该使用';'

注：在for循环头中使用','分割语句是可以的，使用空语句也是可以的，但是一定要保证有2个';'

常见编译错误06

```c
#include <stdio.h>
int main(void) {
    if (1) ;
        printf("In if statment");
    else // error line
        printf("In else statment");
    return 0;
}
```

```
test.c:5:5: error: expected expression
```

错误原因：if-else结构不匹配

注：在该代码段中，因为if后有一条空语句且未加花括号指定作用域，导致else语句被第四行的printf语句隔开了

常见运行时错误01

```c
#include <stdio.h>
int main(void) {
    for (int i = 0; i < 100; ++i) ;
        printf("In for loop?");
    return 0;
}
```

错误原因：for循环未使用花括号指定作用域，并且其后跟了空语句，导致printf语句未能实现循环输出。

常见运行时错误02

```c
#include <stdio.h>
int main(void) {
    int array[10];
    for (int i = 0; i <= 10; ++i) {
        array[i] = i;
    }
    return 0;
}
```

```
Segmatation Fault
```

错误原因：数组下标访问越界，未分配的内存空间进行写操作，导致段错误。

思考：如果是读操作，会发生什么？

常见运行时错误03

```c
#include <stdio.h>
int main(void) {
    int a = 2;
    if (a = 1) {
        printf("a = 1");
    }
    return 0;
}
```

```
a = 1
```

错误原因：错把’=’运算符用在if语句的条件中

常见运行时错误04

```c
#include <stdio.h>
int my_function(i) {
    if (i == 1) {
        return 1;
    }
}
int main(void) {
    my_function(0);
    return 0;
}
```

错误原因：有返回值的自定义函数在某些情况下未返回值

C语言代码规范——一个例子

```c
#include <stdio.h>
int fib(int ); // 函数声明位于全局作用域，进入新的作用域缩进+4空格
int main(void) { // 花括号与括号之间添加空格
    for (int i = 1; i <= 10; ++i) { // 行内分号、逗号后跟空格
        printf("No. %d fib number is %d.\n", i, fib(i));
    }
    return 0;
}
int fib(i) { // 函数定义位于main函数之后
    if ((i == 1) || (i == 2)) { // 运算符加括号以确保优先级正确
        return 1; // 即使if、for仅包含单行语句，也要加花括号
    } else { // 运算符前后加空格
        return fib(i - 1) + fib(i - 2);
    }
}
```

补充例题

```c
#include <stdio.h>
int main(void ) {
    int a, b, c;
    int d, e, f;
    a = 2, b = 3, c = 5;
    d = 0, e = 0, f = 0;

    d = ++a || ++b && ++c;
    e = a >= b ? a : b > c ? b
                           : c;
    f = c > (a > b ? a : b) ? c : (a > b ? a : b);

    d = ++a || (++b && !++c);
    e = a >= b ? a : (b > c ? b : c);
    f = ++b && !++c;
    f = !++b && !++c;

    printf("%f\n", a);
    printf("%f\n", (float)(a));

    return 0;
}
```

补充例题–解析

```
// a = 2, b = 3, c = 5, d = 0, e = 0, f = 0;

d = ++a || ++b && ++c;
```

本题考点：++a与a++的区别，短路运算，逻辑运算符（左结合）

分析：运算符顺序++ > && > ||，先算++a，得到其不为0（逻辑假），短路计算不再计算后续表达式，d=1（逻辑真）。

```
// a = 3, b = 3, c = 5, d = 1, e = 0, f = 0;
```

补充例题–解析

```
// a = 3, b = 3, c = 5, d = 1, e = 0, f = 0;

e = a >= b ? a : b > c ? b
                       : c;
```

本题考点：三目运算符

分析：运算符顺序>= > ?, :

源代码等价于右侧代码

```
if (a >= b) {
    e = a;
} else {
    if (b > c) {
        e = b;
    } else {
        e = c;
    }
}
```

```
// a = 3, b = 3, c = 5, d = 1, e = 3, f = 0;
```

补充例题–解析

```
// a = 3, b = 3, c = 5, d = 1, e = 0, f = 0;

f = c > (a > b ? a : b) ? c : (a > b ? a : b);
```

本题考点：三目运算符

分析：运算符顺序> > ?, :

源代码等价于右侧代码

```
if (c > max(a, b)) {
    f = c;
} else {
    f = max(a, b);
}
```

```
// a = 3, b = 3, c = 5, d = 1, e = 3, f = 5;
```

补充例题–解析

```
// a = 3, b = 3, c = 5, d = 1, e = 3, f = 5;

d = ++a || (++b && !++c);
```

本题考点：++a与a++的区别，短路运算，逻辑运算符（左结合）

分析：运算符顺序++ > ! > && > ||，先算++a，得到其不为0（逻辑假），短路计算不再计算后续表达式，d=1（逻辑真）。

```
// a = 4, b = 3, c = 5, d = 1, e = 3, f = 5;
```

补充例题–解析

```
// a = 4, b = 3, c = 5, d = 1, e = 3, f = 5;

e = a >= b ? a : (b > c ? b : c);
```

本题考点：三目运算符

分析：运算符顺序>= > ?, :

源代码等价于右侧代码

```
if (a >= b) {
    e = a;
} else {
    if (b > c) {
        e = b;
    } else {
        e = c;
    }
}
```

```
// a = 4, b = 3, c = 5, d = 1, e = 4, f = 5;
```

补充例题–解析

```
// a = 4, b = 3, c = 5, d = 1, e = 4, f = 5;

f = ++b && !++c;
```

本题考点：++a与a++的区别，短路运算，逻辑运算符（左结合）

分析：运算符顺序++ > ! > &&，先算++b，得到其不为0（逻辑假），短路计算再计算后续表达式，f=0（逻辑假）。

```
// a = 4, b = 4, c = 6, d = 1, e = 4, f = 0;
```

补充例题–解析

```
// a = 4, b = 4, c = 6, d = 1, e = 4, f = 0;

f = !++b && !++c;
```

本题考点：++a与a++的区别，短路运算，逻辑运算符（左结合）

分析：运算符顺序++ > ! > &&，先算!++b，得到其为0（逻辑假），短路计算不再计算后续表达式，f=0（逻辑假）。

```
// a = 4, b = 5, c = 6, d = 1, e = 4, f = 0;
```

补充例题–解析

```c
// a = 4, b = 5, c = 6, d = 1, e = 4, f = 0;

printf("%f\n", a);
printf("%f\n", (float)(a));
```

本题考点：IEEE754浮点数标准、库函数实现

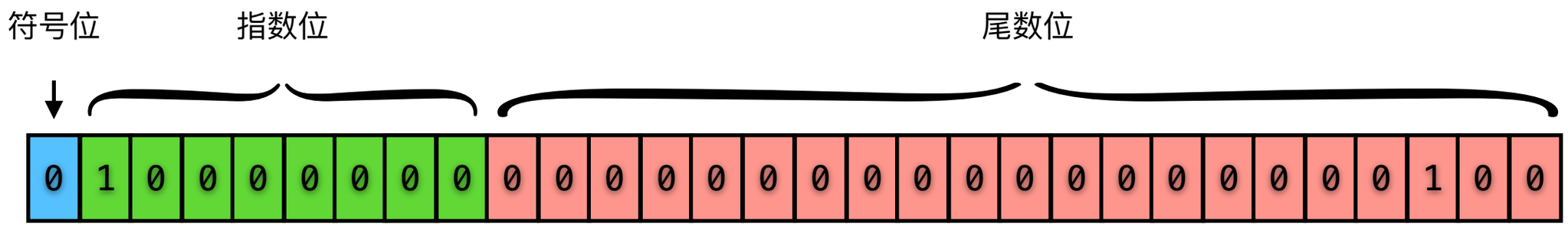
分析：

第一条语句不进行强制类型转换，按照IEEE754标准解释整型变量a，此时输出0（对所有机器都这样的吗？）

第二条语句强制类型转换会将整型变量a转换成浮点数a，此时输出4

```c
// 0.000000
// 4.000000
```

补充例题–IEEE754



$$n = sign \times 2^{(exponent-127)} \times fractions$$

形式	指数	小数部分
零	0	0
非规约形式	0	非0
规约形式	1到 2^{254}	任意
无穷	2^{255}	0
NaN	2^{255}	非零

规约形式浮点数：小数部分为1.frac

非规约形式浮点数：小数部分0.frac

补充例题–解析

```c
printf("%f\n", a); // a = (int )4;
```

在32位机器中：printf函数会首先将输入变量a进行类型提升到double（64-bits），如果a是一个整形值（32-bits），则会涉及到对未定义的内存空间读，输出的结果是一个未定义值。（思考：如果a是 long long类型会如何输出？）

在64位机器中：浮点数使用%xmm寄存器进行参数传递，此时输出的结果是xmm寄存器中的原有值。（涉及到x86-calling convention与ABI）