**第五章 测试与调试**

1.软件开发过程：软件需求分析，软件设计，程序编码，软件测试，软件维护。

2.**缺陷**（**bug**）：程序的错误。

3.**测试**（**test**）:找出缺陷的过程。

4.**调试**（**debug**）：去除错误的过程。

5.

6.**黑盒测试**：检查程序是否满足其输入和输出规格说明。

7.**白盒测试**：保证每一行代码都经过某个级别的测试。

插入的用来检测错误的代码。

8.源代码级别调试器：允许程序在可控环境下执行的工具，

9.**断点**：在程序执行过程中被指定的临时停止点。

10.**条件断点**：特定条件成立才暂停于某行。（隔离被怀疑执行有误的情况）

11.**观察点**：在任意特定条件为真的地方停止的点 。

12.**单步**：**从断点开始**，一次处理一条语句。

①**Step Into**：进入函数内部并在函数中单步执行。

②**Step Over**：不进入函数内部，将函数调用作为一步执行完，用于跳过被认为不包含错误的函数。

③**Step Out**：在函数内部一步完成函数调用。

13.防御性程序设计

①初始化所有变量

②写注释

③避免全局变量

④对齐左右大括号

⑤避免假设

⑥注意编译器警告信息

14.**常见低级错误**

(1)错误：if(x**=**’3’) i++; **//等于与赋值**

正确：if(x**==**’3’) i++;

(2)错误：if(**0<=n<=2**) i++; **//连续不等式**

正确：if(**0<=n&&n<=2**) i++;

(3)错误：if(**year==’7’||’2’**) i++; **//多项或**

正确：if(**year==’7’||year==’2’**) i++;

(4)错误：int i=0,j=0;

for(;i<=5;i++){

for(;j<=5;j++){

printf(“%d%d”,i,j);

}

}

正确：int i=0,j=0;

for(;i<=5;i++){

for(;j<=5;j++){

printf(“%d%d”,i,j);

}

**j=0; //内层循环再次初始化**

}

(5)错误：double b=**4**/**3**\*2.1;

正确：double b=**4.0**/**3.0**\*2.1; **//常量计算类型转换**

(6)错误：int main(){

int x[10];

function(**x[10]**); **//传递数组时要用地址**

}

void function(int cox[]){

cox[1]=0;

}

正确：int main(){

int x[10];

function(**x**);

}

void function(int cox[]){

cox[1]=0;

}

(7)错误：int main(){

char string[3];

printf(“%s\n”,**function(string)**);

}

**char \*function(char \*string)**{

char str[3]={1,2,3}

**return str**; **//不可return局部变量指针**

}

正确：int main(){

char string[3];

**function(string);**

printf(“%s\n”,**string**);

}

**void function(char \*string)**{

**string[0]=1; string[1]=2; string[2]=3;**

}

(8)错误：#include<stido.h>

**#define STRING abc**

int main(){

**char str[4]=”STRING”; //宏替换不替换字符串中内容**

}

正确：#include<stido.h>

**#define STRING “abc”**

int main(){

**char str[4]= STRING;**

}

(9)错误：char string[**3**]=”123”; **//要为空字符留一个位置**

正确：char string[**4**]=”123”;

(10)错误：void function(int);

int main(){

int x=2;

**void function(x);**  **//函数引用不用指明类型**

}

void function(int i){

return ++i;

}

正确：void function(int);

int main(){

int x=2;

**function(x);**

}

void function(int i){

return ++i;

}