C++中的重载函数（在C语言中不允许重载函数）：

例：程序func\_test1

void func(int a);

void func(float b);

int main(int argc, char\* argv[])

{

int a = 5;

float f = 3.5;

func(a);

func(f);

return 0;

}

void func(int a)

{

std::cout << a << std::endl;

}

void func(float b)

{

std::cout << b << std::endl;

}

在C++中，重载函数至少在参数个数，参数类型或参数顺序上有所不同，如果只是返回值不同，是构不成重载的。

void func(int a);

int func(int a); // error

C++中的默认参数的函数：

例：程序func\_test2

void func(int a = 0);

void func1(int b = 5) // 只有函数定义，可以带上默认参数

{

++b;

std::cout << b << std::endl;

}

int main(int argc, char\* argv[])

{

int a = 1;

func();

func(a);

func1();

return 0;

}

void func(int a /\* = 0 \*/) // 在函数声明中有默认参数，在定义中就不允许带上默认参数

{

++a;

std::cout << a << std::endl;

}

默认参数的默认值可以是全局变量，全局常量，甚至是一个函数，但不允许是一个局部变量。

void func(int a /\* = 0 \*/)

{

++a;

std::cout << a << std::endl;

void test(int i = a); // error，非法将局部变量作为默认参数

}

C++默认参数的函数与重载函数：

例：程序func\_test2

void func2(int b, int c = 5);

void func2(int b = 3, int c = 6); // error，重定义默认参数

// void func2(int a = 1, int b, int c = 4, int d = 5); // error，默认参数必须从右至左依次定义

void func2(int a, int b = 1, int c = 4, int d = 5)

{

int sum = a + b + c + d;

std::cout << sum << std::endl;

}

int a = 1;

int b = 3;

func2(a); // a = 1, b = 1, c = 4, d = 5

func2(a, b); // a = 1, b = 3, c = 4, d = 5

func2(a, b, 2, 6); // a = 1, b = 3, c = 2, d = 6

// func2(a, , 2, 4); // error，只能从右至左匹配默认参数

C++中的内联函数：

使用内联函数的原因是提高程序的运行效率，C++中调用函数需要建立栈内存环境，参数传递，产生程序执行转移等，对于一些调用频繁，本身代码比较短的函数可以定义成内联函数，这样程序调用该函数时，会直接用其函数体的代码来替换，提高运行效率。

例：程序func\_test3

inline int func(int a); // 声明为inline后，在定义处可以没有，这一点不同于const函数

// 如果在声明时没有inline关键字，即使在定义时加上，也不会被当成内联函数。

int main(int argc, char\* argv[])

{

int a = 1;

a = func(a);

return 0;

}

int func(int a)

{

++a;

return a;

}

内联函数可以直接定义在头文件中。