§14. C知识补充

4. 条件编译

4.1. 问题的提出

例1：上学期作业

不同编译器函数不同

§. 基础知识题 - 字符数组的输入与输出

4. 多个字符串的输入

★ 从键盘输入含空格字符串的方法(不同编译器不同)

- VS : 有gets_s，无gets， 有fgets
- Dev C++ : 有gets， 无gets_s，有fgets
- Linux : 无gets， 无gets_s，有fgets
- fgets函数的原型定义为：

  fgets(字符数组名，最大长度，stdin);

  但与gets/gets_s的表现有不同，请自行观察

★ scanf/cin通过某些高级设置方式还是可以输入含空格的字符串的，本课程不再讨论

问题：如何让一段存在差异的代码在多编译器下均通过？

§14. C知识补充

4. 条件编译

4.1. 问题的提出

例2：

输入成绩，根据分数
打印及格或不及格

```
if (...) {
    ...
    }
else {
    ...
    }
```

某个游戏软件，适应不同
大小的屏幕

```
if (屏幕是1920*1080) {
    ...
    }
else {
    ...
    }
```

输入成绩，根据分数
打印分数等级

```
if (...) {
    ...
    }
else if (...) {
    ...
    }
else {
   ...
   }
```

可能else if
会重复多次

某个游戏软件，适应不同
大小的屏幕

```
if (屏幕是1920*1080) {
    ...
    }
else if (屏幕是1280*720){
    ...
    }
else {
    ...
    }
```

可能else if
会重复多次

左右的区别：

输入成绩，根据分数打印分数等级 对于if-else形式的双分支(包括if-elseif-else形式的多分支)， 每次执行时只能选择其中的某一个分支执行但多次执行时可能每个分支都可能会被执行到， 因此每个分支都是有意义的	某个游戏软件，适应不同大小的屏幕 对于if-else形式的双分支(包括if-elseif-else形式的多分支)， 对于某种型号的设备，多次执行的都是其中的一个固定分支， 因此分支中的其他部分对某种型号设备的可执行文件而言是无意义的

因为多种型号设备的公用代码很多(非显示部分)，因此希望维护一套源程序代码

既希望维护一套源代码，
又希望可执行文件中
仅包含有效部分(节约空间)

§14.C知识补充

4.条件编译

4.2.问题的引入

引入 ：某些程序在调试、兼容性、平台移植等情况下希望通过简单地设置参数就能生成不同的软件

方法1：把所有可能用到的代码都写进源程序文件中，再全部编译到可执行文件中，执行时根据相应的条件选择不同的代码执行
（分支语句方式，源程序统一，可执行代码大）

方法2：把所有可能用到的代码都写进源程序文件中，在编译之前根据需要选择待编译的代码段，再进行编译，可执行文件中只包含需要的代码段
（条件编译方式，源程序统一，可执行代码小）

§14. C知识补充

4. 条件编译

4.3. 条件编译的三种形式

#ifdef 标识符 　程序段1 #else 　程序段2 #endif	#ifndef 标识符 　程序段1 #else 　程序段2 #endif	#if 表达式 　程序段1 #else 　程序段2 #endif
若定义了标识符 则编译程序段1 否则编译程序段2	若未定义标识符 则编译程序段1 否则编译程序段2	若表达式为真 则编译程序段1 否则编译程序段2

★ 该过程在编译阶段完成，最终形成的可执行文件中只包含两个程序段中的某一个

- # xxx 是C/C++约定的编译预处理指令(例：#define/#include/...)

★ 预编译指令中的表达式与C语言本身的表达式基本一致，逻辑运算、算术运算等均可用于预编译指令

★ 修改预编译条件后，每次都要重新编译链接才能生成新的可执行文件

- 修改条件编译的方法一般是手动修改，再次编译
- 更进一步的方法：1、将条件编译开关放在设置或makefile文件中，通过修改makefile来编译
  2、利用标识不同编译器的预置宏定义来区分不同编译器

★ 形式如if-elseif的多分支条件编译请自行学习

★ 不可能通过执行程序时输入某值或根据程序执行时是否满足某条件的方式来选择，因为是"编译预处理"

程序如下，假设两次的输入为7，-3，写出程序的运行结果

```
#define PROGRAM
main()
{  int t;
   cin >> t;
#ifdef PROGRAM
   if (t>=0)
      cout << "t是非负整数" << endl;
#else
   if (t<0)
      cout << "t是负整数" << endl;
#endif
   cout << "End." << endl;
   return 0;
}
```

注：#ifdef 只判断是否定义，不判断定义值的T/F

未被编译进可执行文件中

输入为7时：
t是非负整数
End

输入为-3时：
End

程序如下，假设两次的输入为7，-3，写出程序的运行结果

```
#define PROGRAM
main()
{  int t;
   cin >> t;
#ifndef PROGRAM
   if (t>=0)
      cout << "t是非负整数" << endl;
#else
   if (t<0)
      cout << "t是负整数" << endl;
#endif
   cout << "End." << endl;
   return 0;
}
```

注：#ifndef 只判断是否定义，不判断定义值的T/F

未被编译进可执行文件中

输入为7时：
End

输入为-3时：
t是负整数
End

程序如下，假设两次的输入为7，-3，写出程序的运行结果

```
#define PROGRAM 0
main()
{  int t;
   cin >> t;
#if PROGRAM
   if (t>=0)
      cout << "t是非负整数" << endl;
#else
   if (t<0)
      cout << "t是负整数" << endl;
#endif
   cout << "End." << endl;
   return 0;
}
```

注：#if 要判断定义值的T/F，未定义则按F处理

未被编译进可执行文件中

输入为7时：
End

输入为-3时：
t是负整数
End

程序如下，假设两次的输入为7，-3，写出程序的运行结果

```
#define PROGRAM 1
main()
{  int t;
   cin >> t;
#if PROGRAM
   if (t>=0)
      cout << "t是非负整数" << endl;
#else
   if (t<0)
      cout << "t是负整数" << endl;
#endif
   cout << "End." << endl;
   return 0;
}
```

注：#if 要判断定义值的T/F，未定义则按F处理

未被编译进可执行文件中

输入为7时：
t是非负整数
End

输入为-3时：
End

§14.C知识补充

4.条件编译

最初的问题：如何让下面程序在多编译器中均能通过

提示：去找各编译器的预置标识

```
int main()
{
    char a[10];
    #if ***   //如果是Dev
       gets(a);
    #elif *** //如果是VS
       gets_s(a);
    #elif *** //如果是Linux
       fgets(a, 10, stdin);
    #endif
    cout << a << endl;
    return 0;
}
```

```
#if ***   //如果是VS
   gets_s(a);
#elif *** //如果是Linux
   fgets(a, 10, stdin);
#elif *** //如果是Dev
   gets_s(a);
#endif
```