代码规范

佳木斯大学acm小组--ldc

换行：

必须换行

定义函数之前

除了小括号内的分号后

左大括号后

右大括号前（除列表形式进行初始化的时候）

右大括号后（除struct ,enum的定义体后，else前　do-while的 while前，初始化列表）

单行超过80字符的最后一个token前

不换行导致不符合语法（宏定义后）

选择换行

单行的代码块之前

左大括号前

列表形式进行初始化时的右大括号前

部分右大括号后（仅struct， enum的定义后 else前 do-while的 while前）

switch的 case和 default的冒号后

enum　定义的逗号后

两块逻辑不相关的代码段之间

建议不换行

连续的两个空行如果出现可以减少一个换行

正常的两个关键词间未被建议换行的地方

例子

修改前

```c
#include <stdio.h>
int main()  {
  int n;char b;
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
      scanf("%c", &b);
      if (b >= 'A' && b <= 'Z') printf("%c",b);
      else printf("NO\n");
  }
  return 0;
}
```

修改后

```c
#include <stdio.h>

int main()  {
  int n;
  char b;                    //换行

  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
      scanf("%c", &b);
      if (b >= 'A' && b <= 'Z') {  //选择换行  视情况加括号
           printf("%c",b);
      } else {                    //选择换行  视情况加括号
         printf("NO\n");
      }
  }

  return 0;
}
```

注释

在C语言中，一共有两种注释的方式。

第一种是我们已经看到的在某一行中插入//的形式，这行中，// 后的所有内容将在程序编译时被忽略，成为仅被程序员所关注的内容。我们将这种形式的注释称为行注释(line comment)。

在C语言中，还有另一种注释以 /* 作为开头，以 */ 作为结尾。在一对 /* 与 */ 之间的所有内容都会被作为注释的内容，只让程序员作为参考，而在程序编译时不被关注。我们将这种形式的注释称为块注释(block comment)。

选择块注释的时候

1.需要把一大段代码暂时性注释掉的时候 。

2.使用一些会读取代码中块注释来生成文档的工具的时候

例子：

修改前：

```
printf("hello whrld");/* 你好世界  */

//for (int i = 0; i <= 5; i++){
//    sum += i;
//}
```

修改后：

```
printf("hello whrld");//你号世界

/*
for (int i = 0; i <= 5; i++){
    sum += i;
} */
```

缩进

需要缩进

if ,else，switch, for ,while,do等关键字后用于组织代码块的大括号内(或这些关键字后单行代码块前) 定义函数的大括号内 goto, case，default等关键字的冒号后组织的一系列语句 struct，enum，union类型定义时和变量初始化时所用的大括号内 如果你在遇到上述情况的时候进行缩进，你会获得一个看起来舒服很多的程序。

例子：

修改前：

```
#include <stdio.h>

int main()  {
  int n;
  char b;

  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
  scanf("%c", &b);
  if (b >= 'A' && b <= 'Z') {
  printf("%c",b);
  } else {
   printf("NO\n");
  }
  }

  return 0;
}
```

修改后：

```c
#include <stdio.h>

int main()  {
  int n;
  char b;

  scanf("%d", &n);
  for (int i = 0; i < n; i++) {          //缩进
      scanf("%c", &b);
      if (b >= 'A' && b <= 'Z') {        //缩进
           printf("%c",b);
      } else {                           //缩进
         printf("NO\n");
      }
  }

  return 0;
}
```

空格

需要空格

+.,>,==.1，&&等前后都需要参加运算的运算符号前后 if， switch, for ,while等关键字、函数定 义名和之后的左小括号之间 不在行尾的逗号、分号之后 不加空格会导致不符合语法的情况下(例如return之后)

选择加空格:

左大括号前、右大括号后

include和头文件、被引入文件之间非运算符冒号之后

再有一点，就是不要额外的写空格: return　　0;这中间连续多出来的空格就是不合适的，我们应该只保留一个空格retucn 0;

例子:

修改前:

```c
#include <stdio.h>

int main(){
  int n,sum;

  scanf("%d",&n);
  for(int i=0;i<=n;i++) {
      sum+=i;
  }

  printf("%d\n",sum);

  return 0;

}
```

修改后：

```
#include <stdio.h>

int main()  {                          //空格
  int n,  sum;

  scanf("%d", &n);
  for (int i = 0; i <= n; i++) {    //空格
      sum += i;                     //空格
  }

  printf("%d\n", sum);              //空格

  return 0;
}
```

总结

规范代码是十分有必要的，简单来说，空格可以让你的代码看起来简洁不拥堵，缩进与换行可以让你的代码看起来调理清晰，层次明显，便于改错。加上合理的运用注释，可以让你的代码易读，便于纠错和维护。