**《C 语言程序设计》**

**（第二版）**

**教案**

**北京出版社**

**课时分配表**

|  |  |  |  |
| --- | --- | --- | --- |
| **章序** | **课程内容** | **课时** | **备注** |
| **1** | **认识 C 程序** | **6** |  |
| **2** | **C 程序设计基础** | **7** |  |
| **3** | **顺序结构程序设计** | **5** |  |
| **4** | **选择结构程序设计** | **3** |  |
| **5** | **循环结构程序设计** | **6** |  |
| **6** | **数组** | **5** |  |
| **7** | **用户自定义函数** | **7** |  |
| **8** | **编译预处理** | **5** |  |
| **9** | **指针** | **9** |  |
| **10** | **构造数据类型** | **5** |  |
| **11** | **文件读写技术** | **6** |  |
| **12** | **C 语言综合实训** | **4** |  |
| **13** | **附录** | **4** |  |
| **总计** |  | **72** |  |

**第8课 编译预处理**

|  |  |  |
| --- | --- | --- |
| **课 题** | 编译预处理 | |
| **课 时** | 5课时（225 min）。 | |
| **教学目标** | **知识技能目标：**  1．了解宏定义。  2．通过学习与练习掌握文件包含。  **思政育人目标：**  让学生通过学习编译预处理，具有积极向上的学习态度，具备自学、自省、自控的能力。 | |
| **教学重难点** | **教学重点：**宏定义  **教学难点：**文件包含 | |
| **教学方法** | 讲授法、问答法、讨论法 | |
| **教学用具** | 电脑、投影仪、多媒体课件、教材 | |
| **教学设计** | 第1节课：考勤（2min）--知识讲解（40min）--作业布置（3min）  第2节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第3节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第4节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第5节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min） | |
| **教学过程** | **主 要 教 学 内 容 及 步 骤** | **设计意图** |
| **考勤**  **（2min）** | ■【教师】清点上课人数，记录好考勤  ■【学生】班干部报请假人员及原因 | 培养学生的组织纪律性,掌握学生的出勤情况 |
| **知识讲解**  （40min） | **【教师】**宏定义  **任务描述**  用带参数的宏计算圆面积。  **预备知识**  在 C 语言源程序中，允许用一个标识符来表示字符串（不是狭义的字符串常量或变量，而是指一串排列符合 C 语言规定的字符），称为“宏”。被定义为“宏”的标识符称为“宏名”。在编译预处理时，对程序中所有出现的“宏名”，都用宏定义中的字符串去代替，这称为“宏代换”或“宏展开”。  宏定义是由源程序中的宏命令完成的，宏代换是由预处理程序自动完成的，宏替换不占用程序的运行时间。“宏”按照有无参数，划分为无参数的宏和带参数的宏两种。  **一、无参数的宏定义**  无参数的宏的宏名后面不带参数，其定义的一般形式为：  #defi ne 标识符 字符串  # 是预处理命令的标志，“defi ne”为宏定义命令，“标识符”为所定义的宏名。“字符串”可以是常数、表达式、格式串等。前面介绍过的符号常量的定义就是一种无参数宏定义。  例如：  #defi ne PI 3.1415926  #defi ne sum 1+2  【注意】  （1）宏定义是用宏名来表示一个字符串，这只是一种简单的代换，字符串中可以包含任何字符，可以是常数，也可以是表达式，预处理程序对它不作任何检查。如果有错误，只能在编译已被宏展开后的源程序里发现。  （2）宏定义不是说明语句，在行末不必加分号，若加了分号，则连分号一起被置换。  例如：  #defi ne sum 1+2;  ……  result=sum/2;  宏展开后实际上就是：  result=1+2;/2  这样显然是不对的。  （3）宏定义必须写在函数之外，其作用域为从宏定义命令起到源程序结束。如果要终止作用域可使用 #undef 命令。例如：  #defi ne PI 3.1415926  void main()  {  …  }  #undef PI  f()  { … }  表示 PI 只在 main 函数中有效，在 f 函数中无效。  （4）宏定义允许嵌套，在宏定义的字符串中可以使用已经定义的宏名。在宏展开时由预处理程序层层代换。  （5）宏名若在源程序中用引号括起来，则预处理程序不对其做宏代换。例如：  #defi ne L 6.283185  void main()  {  printf("L=%f",L);  }  运行结果为：  L=6.283185  本例中可以看到，在 printf 函数中，只有第二个 L 被宏替换了，第一个 L 仍正常显示。这是因为第一个 L 被引号括住，使其不参与替换。  （6）习惯上，宏名用大写字母表示，以便与变量名区别，但也允许用小写字母。  **二、带参数的宏定义**  C 语言中允许宏带有参数。在宏定义中的参数称为形式参数，在宏调用中的参数称为实际参数。对带参数的宏，在调用中不仅要宏展开，而且要用实参去代换形参。  带参数宏定义的一般形式为：  #defi ne 宏名 ( 形参表 ) 字符串  例如：  #defi ne AREA(R) 3.14\*R\*R  #defi ne SUM(X,Y) X+Y  带参数宏调用的一般形式为：  宏名 ( 实参表 );  【例 8-1】把下列运算表达式定义为带参的宏。  （1）求两个数的小者。  （2）求 x 的绝对值。  （3）判断 x 是否奇数。  （4）两个整数交换。  解答：带参的宏定义如下：  （1）#defi ne MIN(a,b) ((a)<(b)?(a):(b))  （2）#defi ne ABS(x) ((x)>=0)?(x):(-x)  （3）#defi ne ISO(x) (((x)%2==1)?1:0)  （4）#defi ne CHANGE(a,b) { int t;t=a;a=b;b=t;}  【注意】  （1）带参数的宏定义中，宏名和形参表之间不能有空格出现，否则会将后面的字符串连空格一起算作无参的宏定义。例如：  #defi ne SUM (X,Y) X+Y /\* SUM 后面有空格 \*/  ……  z=SUM(a,b);  ……  在执行“z=SUM（a，b）；”语句时，经过宏替换后，相当于“z=（X，Y） X+Y （a，b）;”，显然与原本利用宏定义的意图是相违背的。  （2）在带参数的宏定义中，形式参数不分配内存单元，因此不必作类型定义。而宏调用中的实参有具体的值。要用它们去代换形参，因此必须作类型说明。这是与函数中的情况不同的。在函数中，形参和实参是两个不同的量，各有自己的作用域，调用时要把实参值赋予形参，进行值的传递。而在带参数的宏中，只是符号代换，不存在值传递的问题。  （3）在宏定义中的形参是标识符，而宏调用中的实参可以是表达式。例如：  #defi ne PRODUCE(X) X\*X  void main()  {  int a;  a=PRODUCE(3+4);  printf("a=%d",a);  }  此例执行到“PRODUCE（3+4）;”时进行宏替换，替换结果为 a=3+4\*3+4=19。这与函数调用是不同的，函数调用是要把实参表达式的值求出来再赋予形参，而宏替换中对实参表达式不做计算直接按原样替换。  （4）在宏定义中，字符串内的形参通常要用括号括起来以免出错。在上例中假如  宏定义是为了实现求一个数平方的功能，则应该是 7\*7=49，但是不加括号求出来的结果是 19。所以，表达式中的 X 都用括号括起来，结果才是正确的。所以程序应该如下：  #defi ne PRODUCE(X) (X)\*(X)  void main()  {  int a;  a= PRODUCE(3+4);  printf("a=%d",a);  }  （5）带参宏也可以进行嵌套定义。例如：  #defi ne S1(X,Y) X+Y  #defi ne S2(X,Y,Z) (S1(X,Y))\*(S1(Y,Z))\*(S1(Z,X))  ……  C=S2(1,2,3);  ……  最后，“C=S2（1，2，3）;”的宏替换结果为：C=（1+2）\*（2+3）\*（3+1）  =60  **任务实施**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* 程序 : program8\_1.c \*/  /\* 功能 : 用带参数的宏 , 求圆的面积 \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #defi ne AREA(R) 3.14\*R\*R  void main()  {  fl oat r,s;  printf(" 请输入圆的半径 :");  scanf("%f",&r);  s=AREA(r);  printf(" 圆面积是 : %5.2f\n ",s);  }  程序运行结果：  请输入圆的半径：2.0 ↙  圆面积是 :12.56  **【学生】**思考、讨论。 | **展示宏定义，让学生更加仔细的阅读，从而激发学生的学习欲望。** |
| **作业布置**（3min） | **【教师】**布置课后作业  用带参数的宏，求两数之和。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**文件包含  文件包含是 C 语言预处理程序的另一个重要功能，几乎所有的程序都会用到文件包含命令。在前面的程序中已多次用到此命令来包含库函数的头文件。  文件包含命令的一般格式为：  #include < 文件名 >  或  #include " 文件名 "  【功能】把指定的文件插入到该命令行位置，以取代该命令行，从而把指定的文件和当前的源程序文件连成一个源文件。  在程序设计中，文件包含是很有用的，一个大的程序可以分为多个模块，由多个程序员分别编程，这样就充分体现了结构化程序设计的思想，并且可以减少错误，使得纠错更容易。  在使用文件包含时还应注意以下几点：  （1）包含命令中的文件名可以用双引号括起来，也可以用尖括号括起来。例如，以下写法都是允许的：#include <stdio.h> 和 #include“math.h”。但这两种形式是有区别的：  使用双引号则表示首先在当前的源文件目录中查找，若未找到则按系统规定的标准方式检索文件目录；使用尖括号时预处理程序只按系统规定的标准方式检索文件目录，而不在源文件目录中查找。一般而言，对于自己定义的非标准文件使用双引号形式；对于系统提供的标准文件（如 stdio.h、math.h 等），常使用尖括号形式。学生在拿不准的情况下，使用双引号形式是没有错的。  （2）一个 include 命令只能指定一个被包含文件，若有多个文件要包含，则需要多个 include 命令。在多个文件包含时，要注意包含的顺序。  【例 8-2】多个文件的包含举例  f1.c 内容：  int sum(int x,int y)  {  int z;  z=x+y;  return z;  }  f2.c 内容：  int product(int x,int y)  {  int z;  z=x\*y;  return z;  }  f3.c 内容：  int show(int a,int b)  {  printf("input the numbers:");  scanf("%d,%d",&a,&b);  c=sum(a,b)+product(a,b);  return c;  }  假如某文件中要包含这 3 个文件，正确的包含顺序应该是：  #include "f1.c"  #include "f2.c"  #include "f3.c"  假如顺序为：  #include "f3.c"  #include "f2.c"  #include "f1.c"  会发生错误，因为 f3 函数中需要调用 f1 和 f2 中的函数，而由于文件包含的顺序，使得 f3 前面没有这两个函数的定义，从而产生“编译时函数没有定义”的错误。  （3）文件包含允许嵌套，即在一个被包含的文件中又可以包含另一个文件。  **【学生】**思考、讨论。 | **通过教师讲解，了解文件包含的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了文件包含，能够将宏定义做成头文件，并能正确处理文件包含的顺序；让学生知道能够克服自身的弱点，为自己的行为负责，具备良好的个人信誉。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | **【教师】**布置课后作业  以下叙述中不正确的是（　　）。  A. 预处理命令行都必须以 # 号开头  B. 在程序中凡是以 # 开始的语句都是预处理命令行  C. C 程序在执行过程中对预处理命令行进行处理  D. #define IBM\_PC 是正确的宏定义 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】实训课：**词法分析程序  **一、实验目的：**  1.通过设计和实现一个词法分析程序，加深对词法分析原理的理解；  2.掌握在对程序设计语言源程序进行扫描过程中将其分解为各类单词的词法分析方法；  **二、实验内容：**  1.编制一个读单词过程，从输入的源程序中，识别出各个具有独立意义的单词，即基本保留字、标识符、常数、运算符、分隔符五大类。并依次输出各个单词的内部编码及单词符号自身值。（遇到错误时可显示“Error”，然后跳过错误部分继续显示） 词法分析器的功能是输入源程序，输出单词符号。词法分析器的单词符号常常表示成以下的二元式(单词种别码，单词符号的属性值)。  2.保留字：if、then、else、while、do；单词种别码为1;  标识符；单词种别码为2;  无符号整形数；单词种别码为3;  运算符包括：+、-、\*、/、=、<、>、<=、>=；单词种别码为4；分隔符包括：;、(、)； 单词种别码为5;  if (a>10) then b = a + 20;  （1，”if”）  （5，”(”）  （2，”a”）  （4，”>”）  （3，”10”）  （5，”）”）  （1，”then ”）  （2，”b”）  （4，”=”）  （2，”a”）  （4，”+”）  （3，”20”）  （5，”; ”）  **【学生】**思考、讨论。 | **教师通过词法分析程序展示，让学生了解词法分析程序的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了词法分析程序，让学生掌握在对程序设计语言源程序进行扫描过程中将其分解为各类单词的词法分析方法。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **用带参数的宏编写程序，从 3 个数中找出最大数。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】实训课：**展示程序预处理  **一、实验目的**  明确预处理子程序的任务，构造一个简单的预处理子程序，对源程序进行相应的预处理。  **二、实验设备**  1、PC 兼容机一台；操作系统为WindowsWindowsXP。  2、Visual C++ 6.0 或以上版本， Windows 2000 或以上版本，汇编工具（在Software 子目录下）。  **三、实验原理**  定义模拟的简单语言的词法构成，编制读入源程序和进行预处理的程序，要求将源程序读入到文件或存入数组中，再从文件或数组中逐个读取字符进行预处理，包括去掉注释、Tab、Enter和续行符等操作，并显示预处理后的程序。  **四、实验步骤**  1、从键盘读入源程序存放到输入缓冲区中。  2、对源程序进行预处理，预处理后的程序存放到扫描缓冲区中。  3、显示预处理后的程序。  参考源程序（C++语言编写）  //源程序的输入及预处理  #include <fstream.h>  #include <iostream.h>  void pro\_process(char \*);  void main( ) //测试驱动程序  {  //定义扫描缓冲区  char buf[4048]={'\0'}; //缓冲区清0  //调用预处理程序  pro\_process(buf); //在屏幕上显示扫描缓冲区的内容  cout<<buf<<endl;  }  void pro\_process(char \*buf) //预处理程序  {  ifstream cinf("source.txt",ios::in);  int i=0; //计数器  char old\_c='\0',cur\_c;  //前一个字符，当前字符。  bool in\_comment=false;  //false表示当前字符未处于注释中。  while(cinf.read(&cur\_c,sizeof(char))){  //从文件读一个字符  switch(in\_comment){  case false:  if(old\_c=='/'  && cur\_c=='\*'){ //进入注释  i--;  //去除已存入扫描缓冲区的字符'/'  in\_comment=true;  }  else {  if(old\_c=='\\' && cur\_c=='\n')  //发现续行  i--;  //去除已存入扫描缓冲区的字符'\'  else {  if(cur\_c>='A' && cur\_c<='Z')  //大写变小写  cur\_c+=32;  if(cur\_c =='\t' || cur\_c =='\n')  //空格取代TAB换行  cur\_c=' ';  buf[i++]=cur\_c ;  }  }  break;  case true:  if(old\_c=='\*' && cur\_c=='/')  //离开注释  in\_comment=false;  }//end of switch  old\_c= cur\_c;  //保留前一个字符  }//end of while  buf[i++]='#';  //在源程序尾部添加字符'#'  }  **【学生】**思考、讨论。 | **教师通过程序预处理展示，让学生了解程序预处理的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了程序预处理，让学生编制读入源程序和进行预处理的程序，要求将源程序读入到文件数组中。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **如何定义模拟的简单语言的词法构成？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】实训课：**展示LR(1)分析法  **一、实验目的：**  构造LR(1)分析程序，利用它进行语法分析，判断给出的符号串是否为该文法识别的句子，了解LR（K）分析方法是严格的从左向右扫描，和自底向上的语法分析方法。  **二、实验预习提示：**  1.使用LR(1)的优点：  (1)LR分析器能够构造来识别所有能用上下文无关文法写的程序设计语言的结构。  (2)LR分析方法是已知的最一般的无回溯移进-归约方法，它能够和其他移进-归约方法一样有效地实现。  (3)LR方法能分析的文法类是预测分析法能分析的文法类的真超集。  (4)LR分析器能及时察觉语法错误,快到自左向右扫描输入的最大可能。  为了使一个文法是LR的，只要保证当句柄出现在栈顶时，自左向右扫描的移进-归约分析器能够及时识别它便足够了。当句柄出现在栈顶时，LR分析器不需要扫描整个栈就可以知道这一点，栈顶的状态符号包含了所需要的一切信息。如果仅知道栈内的文法符号就能确定栈顶是什么句柄。LR分析表的转移函数本质上就是这样的有限自动机。不过，这个有限自动机不需要根据每步动作读栈，因为，如果这个识别句柄的有限自动机自底向上读栈中的文法符号的话，它达到的状态正是这时栈顶的状态符号所表示的状态，所以，LR分析器可以从栈顶的状态确定它需要从栈中了解的一切。  2.LR分析器由三个部分组成：  (1)总控程序，也可以称为驱动程序。对所有的LR分析器总控程序都是相同的。  (2)分析表或分析函数，不同的文法分析表将不同，同一个文法采用的LR分析器不同时，分析表将不同，分析表又可以分为动作表（ACTION）和状态转换（GOTO）表两个部分，它们都可用二维数组表示。  (3)分析栈，包括文法符号栈和相应的状态栈，它们均是先进后出栈。  分析器的动作就是由栈顶状态和当前输入符号所决定。  LR分析器结构：    **【学生】**思考、讨论。 | **教师通过LR(1)分析法展示，让学生了解LR(1)分析法的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了LR(1)分析法，让学生通过实验更加熟练的操作。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述使用LR(1)的优点。** | 通过课后练习，使学生巩固所学新知识 |
| **教学反思** | 要想促进学生的主动发展，教师在课堂上就要为学生创造成功的环境和条件，这才能让学生充分展示自己，有效地培养学生的自信心，激励学生更加主动地学习。 | |