**中间代码生成程序设计**

基本要求：①掌握中间代码生成的基本方法。

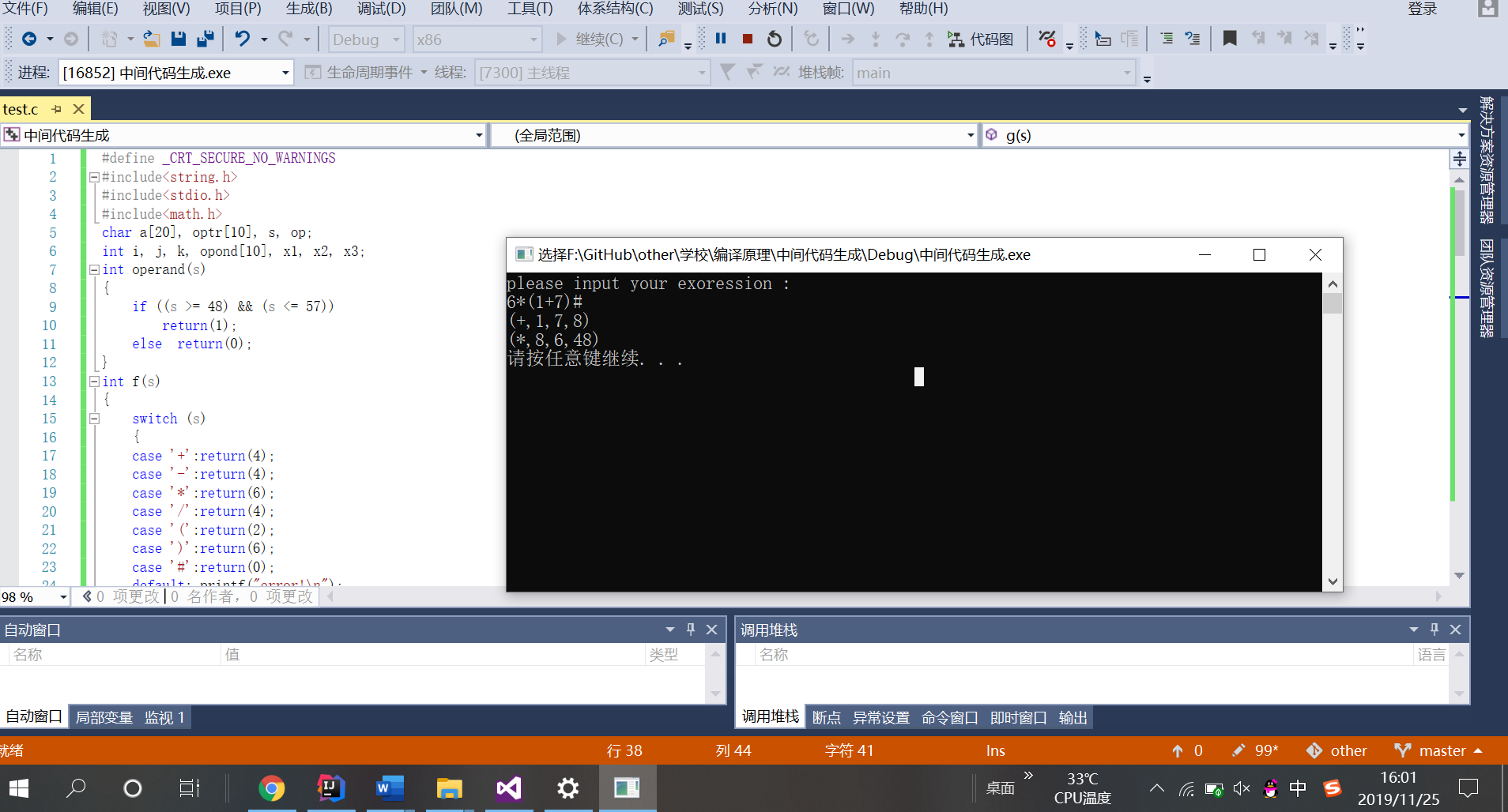
②掌握语法制导翻译模式。

③完成算术表达式的中间代码生成程序。

**一、 算符优先分析法**

算符优先分析法是一种简单且直观的自下而上分析方法，它特别适合于分析程序语言中的各类表达式，并且宜于手工实现。所谓算符优先分析，就是依照算术表达式的四则运算过程来进行语法分析，即这种分析方法要预先规定运算符(确切地说是终结符)之间的优先关系和结合性质，然后借助于这种关系来比较相邻运算符的优先级，以确定句型的“可归约串”来进行归约。因此，算符优先分析法不是一种规范归约，在整个归约过程中起决定性作用的是相继两个终结符的优先关系。

附加语义的方法是采用语法制导翻译的方法，语法制导翻译的方法就是为每个产生式配上一个翻译子程序(称语义动作或语义子程序)，并在语法分析的同时执行这些子程序。语义动作是为产生式赋予具体意义的手段，它一方面指出了一个产生式所产生的符号串的意义，另一方面又按照这种意义规定了生成某种中间代码应做哪些基本动作。在语法分析过程中，当一个产生式获得匹配(对于自上而下分析)或用于归约(对于自下而上分析)时，此产生式相应的语义子程序就进入工作，完成既定的翻译任务。



源代码：

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<string.h>

#include<stdio.h>

#include<math.h>

char a[20], optr[10], s, op;

int i, j, k, opond[10], x1, x2, x3;

int operand(s)

{

if ((s >= 48) && (s <= 57))

return(1);

else return(0);

}

int f(s)

{

switch (s)

{

case '+':return(4);

case '-':return(4);

case '\*':return(6);

case '/':return(4);

case '(':return(2);

case ')':return(6);

case '#':return(0);

default: printf("error!\n");

}

}

int g(s)

{

switch (s)

{

case '+':return(3);

case '-':return(3);

case '\*':return(5);

case '/':return(5);

case '(':return(7);

case ')':return(2);

case '#':return(0);

default:return (0); printf("error!\n");

}

}

void get()

{

s = a[i];

i = i + 1;

}

int main()

{

printf("please input your exoression : \n");

i = 0;

do

{

i = i + 1;

scanf("%c", &a[i]);

} while (a[i] != '#');

i = 1; j = k = 1;

optr[j] = '#';

get();

while (!((optr[j] == '#' && (s == '#'))))

{

if (operand(s))

{

opond[k] = s - 48;

k = k + 1;

get();

}

else if (f(optr[j]>g(s)))

{

op = optr[j];

j = j - 1;

x1 = opond[k - 1];

x2 = opond[k - 1];

k = k - 1;

switch (op)

{

case '+':x3 = x1 + x2; break;

case '\*':x3 = x1\*x2; break;

case '-':x3 = x1 - x2; break;

case '/':x3 = x1 / x2; break;

}

opond[k] = x3;

k++;

printf("%c, %d, %d, %d)\n", op, x1, x2, x3);

}

else if (f(optr[j]<g(s)))

{

j = j + 1;

optr[j] = s;

get();

}

else if (f(optr[j] == g(s)))

{

if (optr[j] == '('&&')')

{

j = j - 1;

get();

}

else printf("error!!!\n");

}

else printf("error!!!\n");

}

}