栈的应用：逆波兰计算器

逆波兰表达式

- 什么是逆波兰表达式？
  - 通常二元运算符总是置于与之相关的两个运算对象之间，这种表示法也称为中缀表示。
  - 波兰逻辑学家**J.Lukasiewicz**于**1929**年提出了另一种表示表达式的方法：每一运算符都置于其运算对象之后——故称后缀表示（逆波兰表达式又称后缀表达式）
  - 例如：
    - $(a*(b-c))+d \longrightarrow a\ b\ c\ -*\ d+$
    - $2-3+5*2*3+4 \longrightarrow 2\ 3\ -5\ 2\ *3\ *\ +4+$

逆波兰表达式计算器

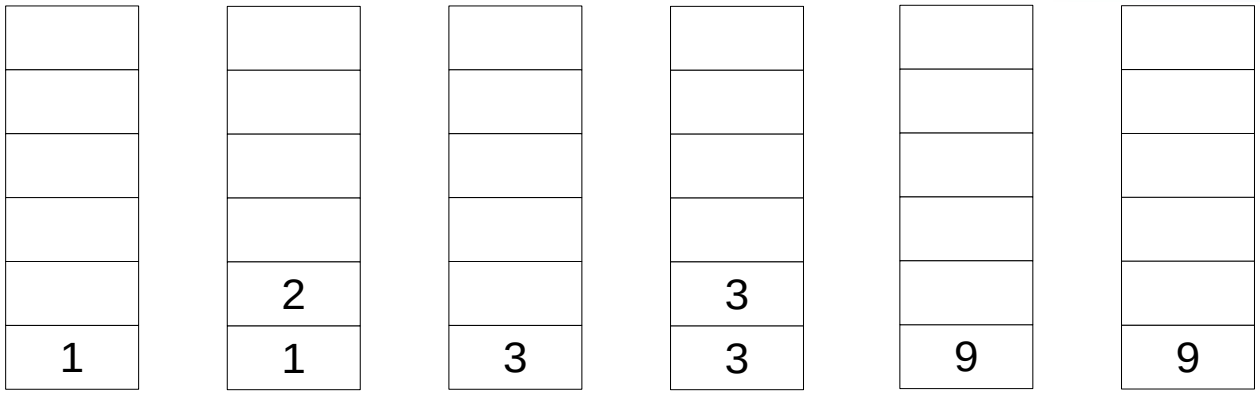
- 要求：
  - 输入：
    - ？ —— 表示后续将读入一个操作数
    - +,-,*,/ —— 表示读入了一个运算符
    - = —— 表示表达式输入结束，需要将求解的结果输出
  - 例如：
    - ?a?b+?c?d +*= 表示求解表达式$(a+b)*(c+d)$的值
    - ?a?b?c −=*?d += 表示先求解a * (b − c)，并输出结果，再读入d，求解$\big(a*(b-c)\big)+d$的值输出

逆波兰表达式计算器

- 算法思想：**程序从键盘接受命令符**
  - 如输入的命令为“**?**”：
    - 则继续读入一个操作数，由于对该操作数执行的运算符还未知，所以暂时将它存储起来；
  - 如输入的是“**+-*/**”等运算符：
    - 则此时应将最近保存过的数拿出两个作算术运算，并将运算的中间结果存储起来
    - 由于最近存储的操作数（或中间结果）即是即将拿出来作运算的操作数，也即后存储的先取出来（**LIFO**），故应把这些操作数存储在栈中
  - 如输入的命令为“**=**”：
    - 将刚才的运算结果即栈顶显示出来

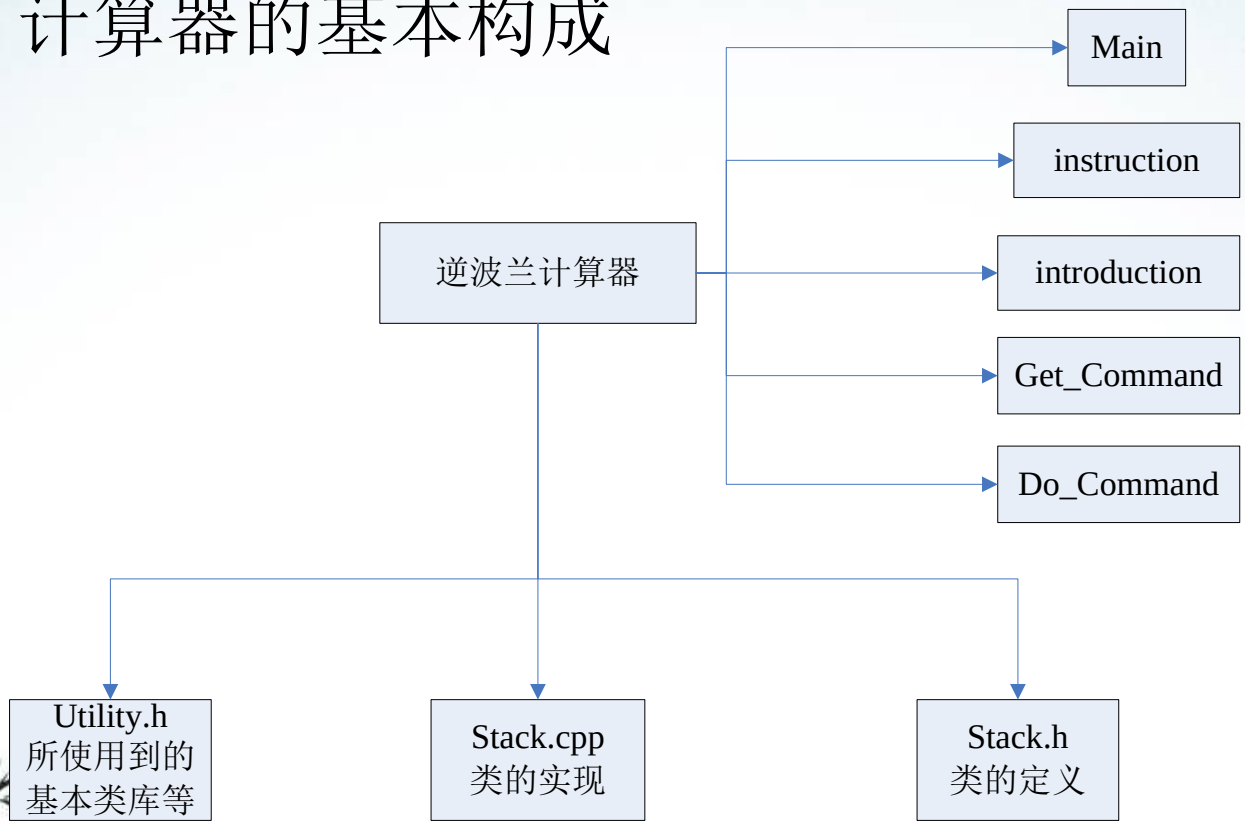
逆波兰表达式计算器

- 求解过程：
  - 输入：?1?2+?3 *=
  - 输出：9
  - 过程：



逆波兰表达式计算器

- 计算器的基本构成



逆波兰计算器——主函数

```
typedef double Stack_entry;
#include “Stack.h"
int main( )
/* Post: The program has executed simple arithmetic commands entered by the user.
Uses: The class Stack and the functions introduction, instructions, do_command,and get_command. */
{
    Stack stored_numbers;
    introduction( );
    instructions( );
    while (do_command(get_command( ), stored_numbers));
}
```

逆波兰计算器——get_command

```
char get_command( ){   /*obtains a command from the user, checking that it is valid and converting it to lowercase*/
    char command;
    bool waiting = true;
    cout << "Select command and press < Enter > :";
    while (waiting) {
        cin >> command;
        command = tolower(command);
        if (command == '?' || command == '=' || command == '+'
        ||command == '-' || command == '*' || command == '/'
        ||command == 'q')
                waiting = false;
        else
                cout << "Please enter a valid command:" << endl<<
        "[?]push to stack [=]print top" <<endl<< "[+] [-] [*] [/] are
        arithmetic operations" << endl<< "[Q]uit." << endl;
    }
    return command;
}
```

逆波兰计算器——do_command

```
bool do_command(char command, Stack &numbers) {
/* Pre: The first parameter specifies a valid calculator command.
Post: The command specified by the first parameter has been applied to the
Stack of numbers given by the second parameter. A result of true is returned
unless command == ' q '.
Uses: The class Stack. */
        double p, q;
        switch (command) {
        case '?':    // read
                cout << "Enter a real number: " << flush;
                cin >> p;
                if (numbers.push(p) == overflow)
                        cout << "Warning: Stack full, lost number" << endl;
                break;
```

逆波兰计算器——do_command(续)

```
case '=':    // print
        if (numbers.top(p) == underflow)
                cout << "Stack empty" << endl;
        else
                cout << p << endl;
        break;
case 'q':
        cout << "Calculation finished.\n";
        break;
```

逆波兰计算器——do_command(续)

```
        case '+':
            if (numbers.top(p) == underflow)
                    cout << "Stack empty" << endl;
            else {  numbers.pop( );
                    if (numbers.top(q) == underflow) {
                            cout << "Stack has just one entry" << endl;
                            numbers.push(p);
                    }else {
                            numbers.pop( );
                            if (numbers.push(q + p) == overflow)
                                    cout<<"Warning: Stack full, lost result“ <<endl;
            } }
            break;
            // Add options for further user commands.
        }
        return true;
}
```

思考

- 表达式变成中缀表达式
  - 括号表示优先级
    - 例如：**(3+4)+(5*2)**
  - 优先级由运算符自身确定
    - 例如：**(3+4)*2+6/3**