Intermediate C Programming

Lesson8

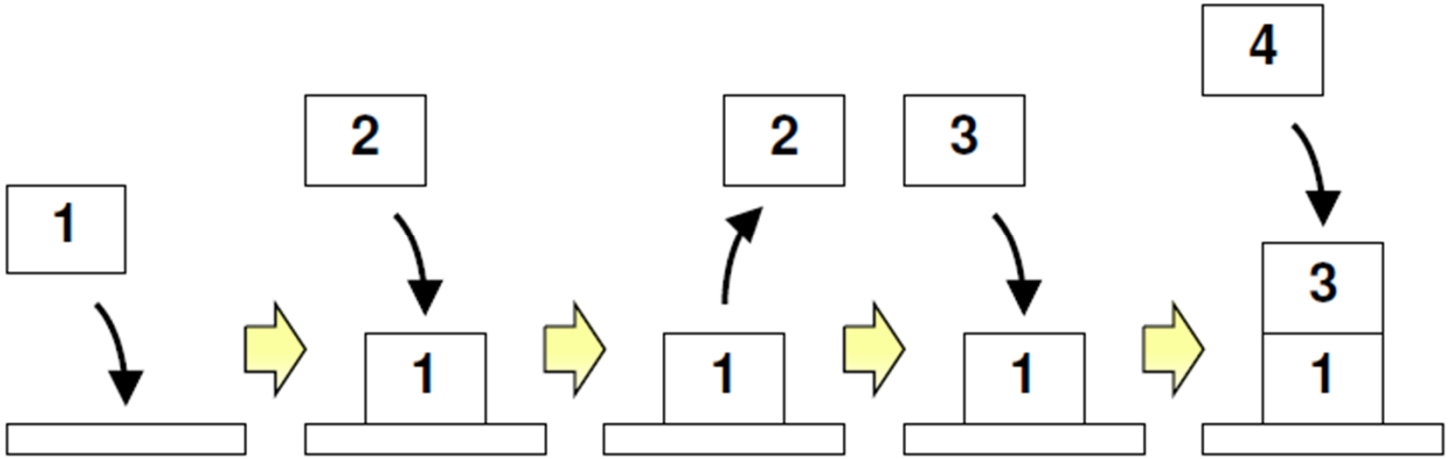
STACK and QUEUE

Today's outline

- STACK
- QUEUE
- Exercise

Stack

- What is the stack?
- Last In First Out（LIFO）



Stack

- Array : The size of array is the number of data in the stack
- Pointer: for stack_top

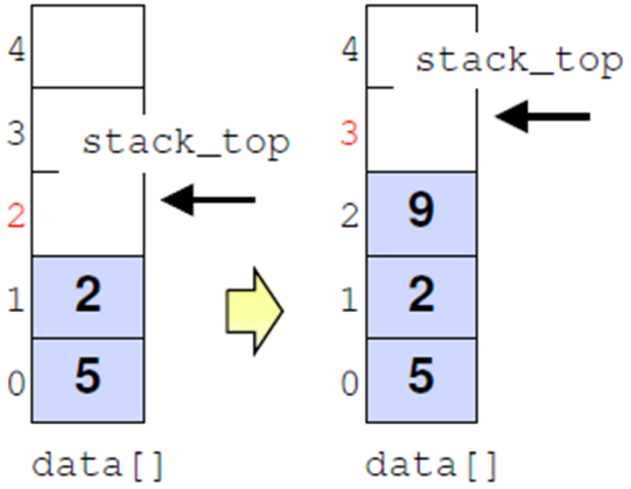
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define STACK_MAX 10
4 double data[STACK_MAX];
5 int stack_top=0;
```

- Global Variables : declared out of the function and can be used by all functions. data[STACK_MAX],stack_top
- Local Variables: declared within the function and can only be used by this function

Stack

- Push: put data in the stack
- Put the data at the top of the stack and update the position of (stack_top)

data[stack_top] = value;
stack_top++;




```
6 void stack_push( double value ){
7     data[stack_top] = value;
8     stack_top++;
9 }
```

Stack

- Pop : pop data from the stack
- Decease stack_top by one

  stack_top--;

```
10 double stack_pop( void ){
11     stack_top--;
12     return data[stack_top];
13 }
```

Stack

- When the stack is full,
- stack overflow

```
5  void stack_push( double value ){
6      if( stack_top == STACK_MAX ){
7          printf( "error: stack overflow\n" );
8          exit(1);
9      }
10     data[stack_top] = value;
11     stack_top++;
12 }
```

Stack

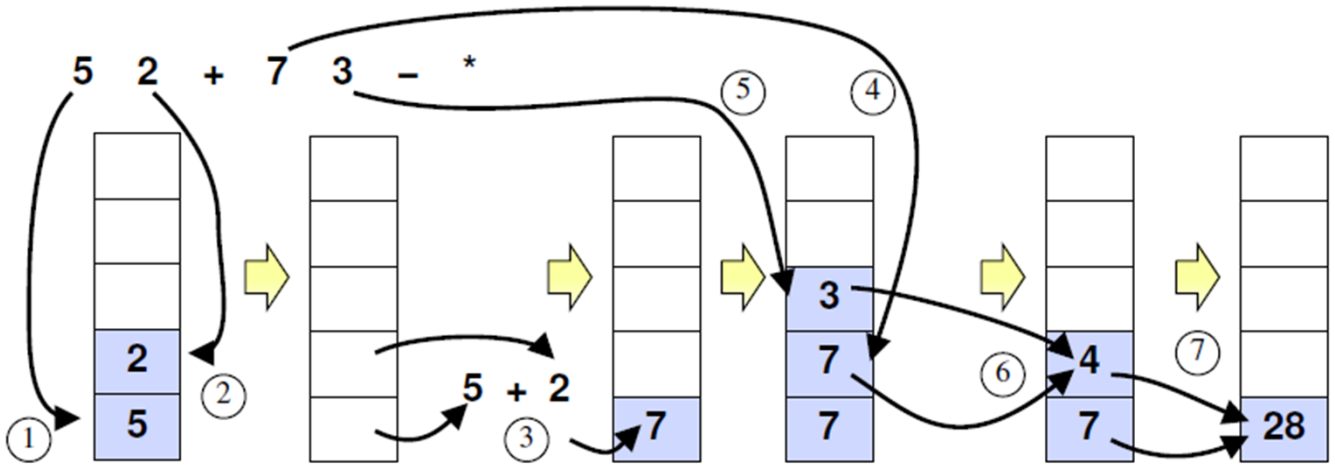
- When the stack is empty

  stack underflow

```
13 double stack_pop( void ){
14     if( stack_top == 0 ){
15         printf( "error: stack underflow\n" );
16         exit(1);
17     }
18     stack_top--;
19     return data[stack_top];
20 }
```

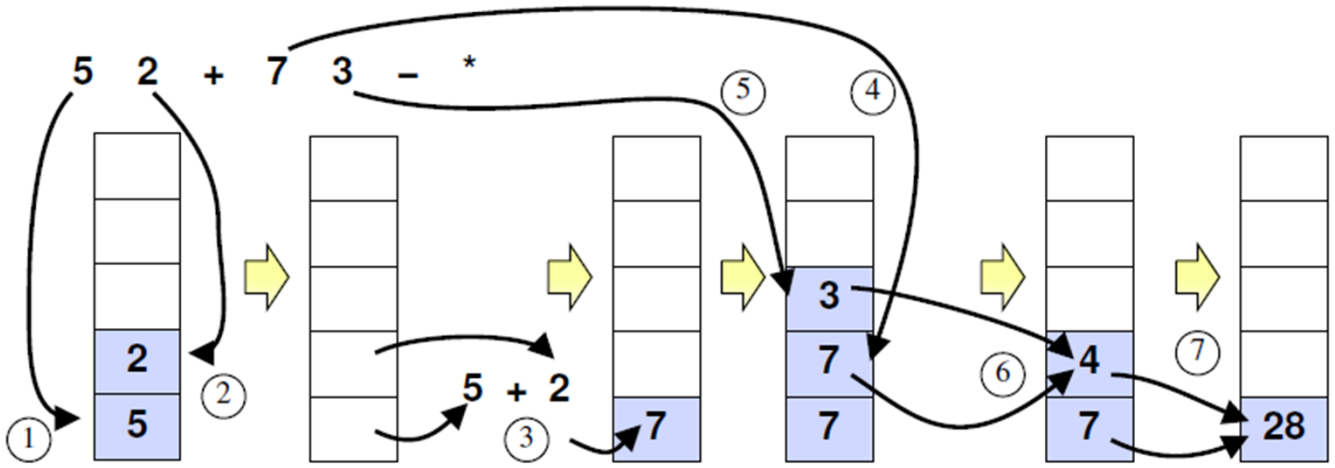
Stack

- Let's calculate (5 + 2) x (7 - 3)
- Reverse Polish Notation: 5 2 + 7 3 - x



Stack

- Let's calculate (5 + 2) x (7 - 3)
- Reverse Polish Notation: 5 2 + 7 3 - x



Stack

```
21 int main(){
22     char buf[10] = "";
23     double x, y, ans;
24     while( buf[0] != '=' ){
25         printf( ">> " );
26         scanf( "%s", buf );
27         if( buf[0] == '+' ){
28         else if( buf[0] == '-' ){
29         else if( buf[0] == '*' ){
30         else if( buf[0] == '/' ){
31         else if( buf[0] == '=' ){
32         else {
33         }
34     }
35     printf( "result = %f\n", ans );
36     return 0;
37 }
```

Stack

- if(buf[0] == '+')

```
y = stack_pop();
x = stack_pop();
stack_push( x + y );
```

- if(buf[0] == '=')

```
ans = stack_pop();
```

- calculation

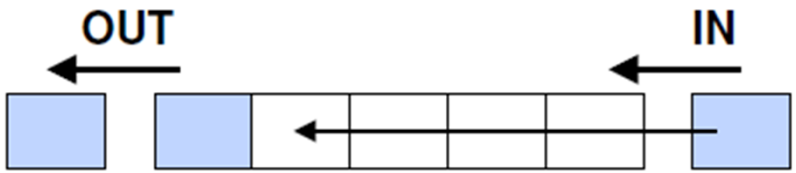
```
x =  atof(buf);
stack_push( x );
```

Exercise

Finish the program of the stack

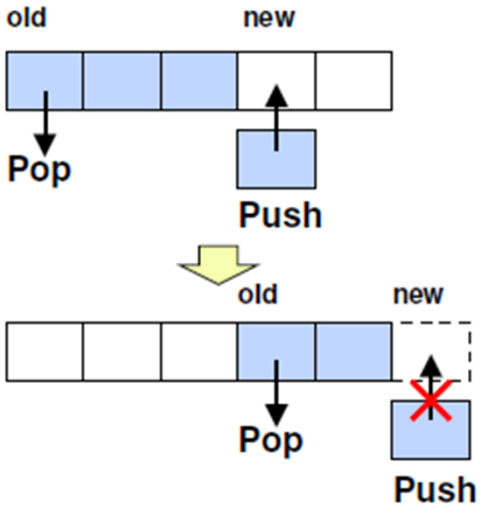
Queue

- What is the queue?
- First In First Out（FIFO）

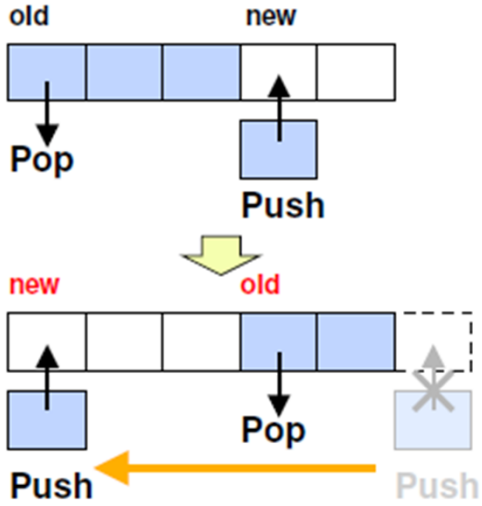


Queue

- Problems of Queueing



- Link Buffer



Queue

- Array : The size of array is the number of data in the queue
- Pointer: for head and tail of the queue

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define QUEUE_MAX 10
4 int queue[QUEUE_MAX+1];
5 int queue_head=0, queue_tail=0;
```

- Global Variables
- Size of queue = the number of data +1
- queue_head=0; queue_tail=0;

Queue

- Push/Enqueue: add data in the queue
- Put the data at the end of the queue and update the position of (queue_tail)

```
6 void enqueue( int data ){
7     queue[queue_tail] = data;
8     queue_tail = (queue_tail + 1) % (QUEUE_MAX + 1);
9 }
```

Queue

- Pop/Dequeue: read/take data from the queue
- take the first data in the queue and update the position of (queue_head)

```
10 int dequeue( void ){
11     int data;
12     data = queue[queue_head];
13     queue_head = (queue_head + 1) % (QUEUE_MAX + 1);
14     return data;
15 }
```

Queue

- When the queue is empty,

```
6  void enqueue( int data ){
7      if( (queue_tail+1)%(QUEUE_MAX+1) == queue_head ){
8          printf( "error: no more space in queue\n" );
9          exit(1);
10     }
11     queue[queue_tail] = data;
12     queue_tail = (queue_tail + 1) % (QUEUE_MAX + 1);
13 }
```

Queue

- When the queue is full

```
14 int dequeue( void ){
15     int data;
16     if( queue_head == queue_tail ){
17         printf( "error: empty queue\n" );
18         exit(1);
19     }
20     data = queue[queue_head];
21     queue_head = (queue_head + 1) % (QUEUE_MAX + 1);
22     return data;
23 }
```

Exercise

Finish the program of the queue

```
24 int main(){
25     int op=0, value;
26     while( op != 3 ){
27         printf( "Select 1:enqueue 2:dequeue 3:quit\n" );
28         scanf( "%d", &op );
29         if( op == 1 ){
30             printf( "value? > " );
31             scanf( "%d", &value );
32             enqueue( value );
33         } else if( op == 2 ){
34             value = dequeue();
35             printf( "value = %d\n", value );
36         }
37     }
38     return 0;
39 }
```