Computer Programming 1 Lab

2022-12-01

andyjjrt 洪晙宸

Outline

- String functions
  - strcpy
  - strcmp
- Struct
- Stack
- Queue
- Exercise 9

strcpy

```
char* strcpy( char* dest, const char* src )
```

```
#include <stdio.h>
#include <string.h>

int main() {
  char src[] = "Hello world";
  char dest[12];
  strcpy(dest, src);
  printf("%s", dest);
}
```

```
$ ./a.out
Hello world
```

⏰ Avoid making src length > dest length, may cause unknown error

strcmp

```
int strcmp( const char* s1, const char* s2 )
```

```
#include <stdio.h>
#include <string.h>

int main() {
  char s1[10] = "Hello";
  char s2[10] = "Hello";
  char s3[10] = "hello";
  printf("%d\n", strcmp(s1, s2));
  printf("%d\n", strcmp(s1, s3));
}
```

```
$ ./a.out
0
-32
```

Different environment may have different return value.

What is String

```c
char name[10] = "Andy Hung";
```

```c
char name[10] = {'A', 'n', 'd', 'y', ' ', 'H', 'u', 'n', 'g', '\0'};
```

Treat a string as a normal array, whatever you do to the array, it can apply to string

Struct

- A set of variables

```c
struct Monster {
  int id;
  char name[10];
  int health;
  int attack;
  int recovery;
  // ...
};
```

```c
struct Team {
  struct Monster monster[5];
  struct Monster helper;
};
```

Struct

Use `typedef`

```c
typedef struct Monster Monster;
typedef struct Team Team;
```

Initialize like a normal variable

```c
Monster monster_1;
monster_1.id = 2480;
strcpy(monster.name, "全知的惡魔 · 拉普拉斯");
monster_1.health = 2158;
monster_1.attack = 2523;
monster_1.recovery = 576;
```

Or using malloc

```c
Monster* monster_1 = malloc(sizeof(Monster));
monster_1 -> id = 2480;
strcpy(monster -> name, "全知的惡魔 · 拉普拉斯");
monster_1 -> health = 2158;
monster_1 -> attack = 2523;
monster_1 -> recovery = 576;
```

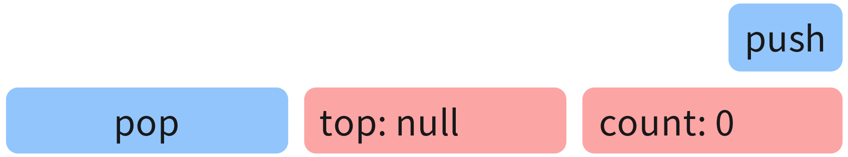
Struct

Construct a team

```
Team team;
team.helper = monster_1;

for(int i = 0; i < 5; i++) {
  team.monster[i] = monster_1
}
```

Stack

First in last out



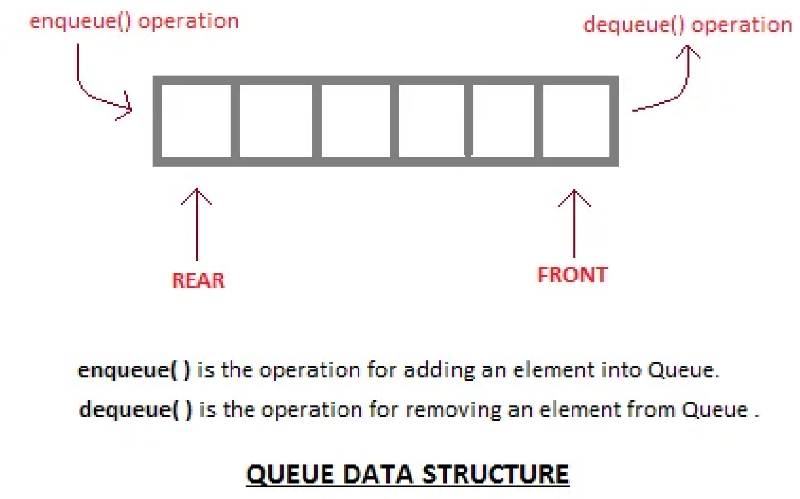
```c
#define SIZE 10
int Stack[SIZE] = {};
int top = -1;

int empty() {
  return top < 0;
}

int full() {
  return top >= SIZE - 1;
}

void push(int val) {
  if(!full()) stack[++top] = val;
}

void pop() {
  if(!empty()) --top;
}

int getTop() {
  return stack[top];
}
```

Queue

First in first out

enqueue

dequeue | first: null | rear: null | count: 0



enqueue( ) is the operation for adding an element into Queue.

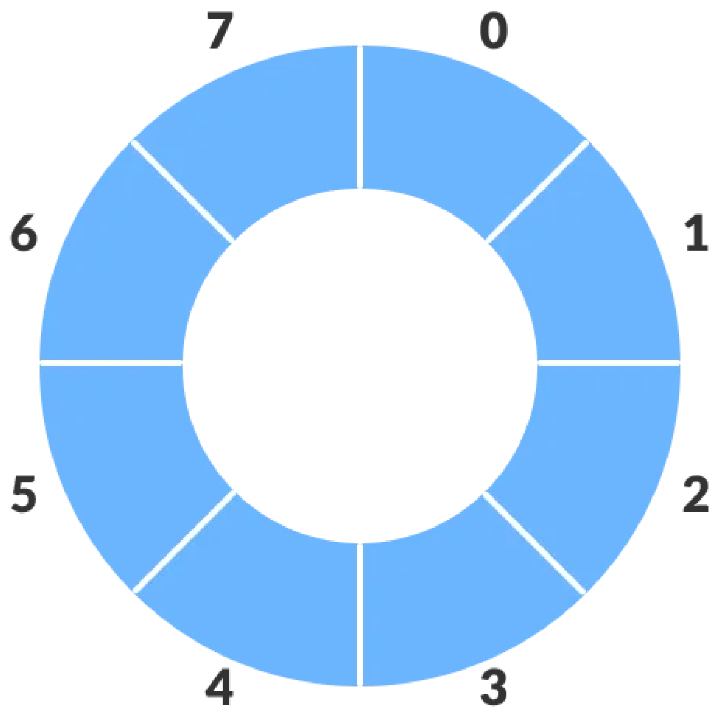
dequeue( ) is the operation for removing an element from Queue .

QUEUE DATA STRUCTURE

Queue

```c
#define SIZE 10
int Queue[SIZE] = {};
int front = 0;
int rear = 0;

int empty() {
  return front == rear;
}

void enqueue(int val) {
  Queue[rear++] = val;
}

void dequeue() {
  front++;
}

int getFront() {
  return Queue[front];
}

int getRear() {
  return Queue[rear];
}
```

Queue



```
#define SIZE 10
int Queue[SIZE] = {};
int front = 0;
int rear = 0;

int empty() {
  return front == rear;
}

void enqueue(int val) {
  Queue[rear++ % SIZE] = val;
}

void dequeue() {
  front++;
}

int getFront() {
  return Queue[front % SIZE];
}

int getRear() {
  return Queue[rear % SIZE];
}
```

Exercise 9: 混合字串數字和

每行input會是整數(含負數)和字母的混合字串，請輸出每行出現數字的總和

- Input: 測資包含多行由數字（N 為正負整數）和任意字母的組成的混合字串

```
863QA667kP107jpLjP617G
-619Nri-805vE559z-478S284zs560n
658q-692Z-327HNMJ31Pd-763j-92b
809ZG-307SB459E-82l748XT-120jp
wB-808El-282pqv-542G27sv
```

- Output: 請輸出混合字串中的數字總和

```
2254
-499
-1185
1507
-1605
```

Any questions?