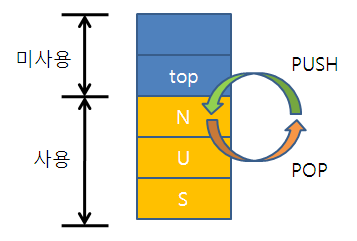
**———————————————————————————————————————————————————**

[1] 기초 문제

———————————————————————————————————————————————————

1. **다음 그림에서 찾을 수 있는 스택의 용어들에 어떤 것이 있는지 나열하시오.**

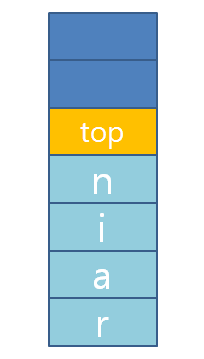


1. **스택 용어와 관련된 문제다. 빈 칸에 들어가는 단어를 맞춰보도록 한다.**
   1. ① LIFO ② 스택 ③ 후입선출
   2. ① top ② 꼭대기 ③ 마지막
   3. ① pop ② 꺼내는, 꺼낸 ③ 마지막
2. **스택을 사용하고 있는 곳을 세 가지씩만 나열해 보자.**
   1. 프링글스, 쇼핑 카트, 크리넥스, 포스트잇, 막힌 길, CD 케이크
   2. 지역 변수, 함수 호출, 물르기, 프로토콜 스택, undo, 미로

|  |
| --- |
| 해답  정답으로 간주할 수 있는 것들이 너무 많아서 이번 장에서 설명한 것만 실었습니다. |

1. **최대 7개의 문자를 저장할 수 있는 스택이 있다. 아래와 같은 입출력이 발생한다면, 꺼낸 데이터를 순서대로 보여주고 최종적인 스택의 내부를 그려라.**

출력 : oitaedore

 [그림 3-82] 스택\_내부상태\_해답

1. **스택을 구현하는 방법으로 배열과 단순 연결 리스트의 두 가지 방법이 있다. 빈 칸에 들어가는 내용을 단답형으로 채워보자.**

|  |  |  |
| --- | --- | --- |
|  | 동적 배열 | 단순 연결 리스트 |
| 크기 | 무제한 | 무제한 |
| 성능 | 좋다 | 나쁘다 |
| 구현 | 쉽다 | 어렵다 |
| 메모리 할당 | 일괄 할당, 일괄 해제 | 매번 할당, 매번 해제 |
| 메모리 사용 | 적다 | 많다 |

|  |
| --- |
| 해답  표에 대한 완전한 설명은 1장의 스택의 종류에 있습니다. 설명이 너무 길어 기본적인 답안만 제공합니다. |

**———————————————————————————————————————————————————**

[2] 기본 실습 문제

———————————————————————————————————————————————————

1. **본문에서 구현했던, 스택을 구성하는 핵심 함수들이다. 직접 구현해 보자.**
   1. void push(char data)

{

g\_array[g\_top++] = data;

}

* 1. char pop(void)

{

return g\_array[--g\_top];

}

* 1. int isEmpty(void)

{

return g\_top == 0;

}

1. **본문에서 배우지 않았거나 구현을 했어도 조금은 어려운 함수들이다. 구현하고 검증까지 하도록 하자.**
   1. void print(void)

{

int i;

for(i = g\_top-1; i >= 0; i--)

printf("%c ", g\_array[i]);

printf("\n");

}

* 1. void clear(void)

{

g\_top = 0;

}

* 1. 이 문제는 4자리에 맞게 공백을 넣는 것이 어려운 문제였습니다.

void convertRadix(int dec)

{

int i = 0;

for( ;dec != 0; dec /= 2)

push(dec%2), i++;

while(isEmpty() == 0)

{

printf("%d", pop());

if(--i%4 == 0)

printf(" ");

}

printf("\n");

}

출력 대신 값을 반환하는 함수도 실어봤습니다. 정수 범위에 영향을 받으므로, 10자리 2진수의 최대값을 가리키는 1023을 넘어서면 잘못된 결과가 나옵니다. 그럴 경우 문자열을 반환하도록 수정할 필요가 있겠습니다.

int convertRadix(int dec)

{

while(dec != 0)

push(dec%2), dec /= 2;

while(isEmpty() == 0)

dec = dec\*10 + pop();

return dec;

}

**———————————————————————————————————————————————————**

[3] 응용 실습 문제

———————————————————————————————————————————————————

1. 코드가 복잡해서 테스트할 수 있는 코드까지 실었습니다.

#include <stdio.h>

#include <string.h>

#include "Stack.h"

char\* parenMatch(char\* str)

{

char\* paren = "{}[]()<>";

char\* p;

for( ; \*str; str++)

{

printf("%c", \*str);

// 괄호라면

if(p = strchr(paren, \*str))

{

// 여는괄호

if((p-paren)%2 == 0)

{

push(\*str);

}

// 닫는괄호

else

{

p = strchr(paren, pop());

if(\*(p+1) != \*str)

{

printf("\n");

return str;

}

}

}

}

printf("\n");

// 스택에남아있으면실패

return isEmpty() ? NULL : str;

}

void main(void)

{

char s[] = "[<You( (are) { so }[ <beautiful> ]to ) me.>]";

char\* p;

printf("%s\n", s);

if(p = parenMatch(s))

printf("실패: %s\n", p);

else

printf("성공\n");

}

1. 본문처럼 세 개의 파일로 나눴습니다. 최대한 본문과 비슷하게 꾸몄습니다. 테스트를 용이하게 하기 위해 스택의 크기는 5로 잡았고, main 함수에서 push와 pop 함수를 번갈아 호출했습니다.

**// DOS.h**

#pragma once

void push(char data);

char pop(void);

int isEmpty(void);

int isFull(void);

void print(void);

**// DOS.c**

#include <stdio.h>

#include "Stack.h"

enum { STACK\_SIZE = 5 };

static char g\_array[STACK\_SIZE];

static int g\_top = 0, g\_bottom = 0;

void push(char data)

{

g\_array[g\_top] = data;

g\_top = (g\_top+1)%STACK\_SIZE;

if(g\_top == g\_bottom)

g\_bottom = (g\_bottom+1)%STACK\_SIZE;

}

char pop(void)

{

return g\_array[g\_top = (g\_top+STACK\_SIZE-1)%STACK\_SIZE];

}

int isEmpty(void)

{

return g\_top == g\_bottom;;

}

int isFull(void)

{

return (g\_top+1)%STACK\_SIZE == g\_bottom;;

}

void print(void)

{

int i;

printf("\*\*%d : %d \*\* ", g\_bottom, g\_top);

for(i = g\_bottom; i != g\_top; i=(i+STACK\_SIZE+1)%STACK\_SIZE)

printf("[%d] %c ", i, g\_array[i]);

printf("\n");

}

**// main.c**

#include <stdio.h>

#include "Stack.h"

void main(void)

{

int i;

for(i = 0; i < 7; i++)

push(i+'A'), print();

printf("\n");

for(; i < 10; i++)

pop(), print();

printf("\n");

for(i = 0; i < 7; i++)

push(i+'A'), print();

printf("\n");

while(isEmpty() == 0)

pop(), print();

printf("\n");

}