|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』실습 보고서 | | | |
| 제목 | 4장 실습( O ) 과제( ) | 제출일자 | 2022.   10.  05 . |
| 학번 | 201911608 | 이름 | 김지환 |

|  |
| --- |
| 실습  1. 프로그램 4.1  #include <stdio.h>  #include <stdlib.h>  #define MAX\_STACK\_SIZE 100  typedef int element;  element stack[MAX\_STACK\_SIZE];  int top = -1;  int is\_empty()  {  return (top == -1);  }  int is\_full()  {  return (top == (MAX\_STACK\_SIZE - 1));  }  void push(element item)  {  if (is\_full()) {  fprintf(stderr, "스택 포화 에러\n");  return;  }  else stack[++top] = item;  }  element pop()  {  if (is\_empty()) {  fprintf(stderr, "스택 공백 에러\n");  exit(1);  }  else return stack[top--];  }  element peek()  {  if (is\_empty()) {  fprintf(stderr, "스택 공백 에러\n");  exit(1);  }  else return stack[top];  }  int main(void)  {  push(1);  push(2);  push(3);  printf("%d\n", pop());  printf("%d\n", pop());  printf("%d\n", pop());  return 0;  }  실행결과  스택은 1 -> 2 -> 3 순서로 쌓이며 출력은 3 -> 2 -> 1의 LIFO 이다. |
| 2. 프로그램 4.2  #include <stdio.h>  #include <stdlib.h>  #define MAX\_STACK\_SIZE 100  #define MAX\_STRING 100  typedef struct {  int student\_no;  char name[MAX\_STRING];  char address[MAX\_STRING];  } element;  element stack[MAX\_STACK\_SIZE];  int top = -1;  int is\_empty()  {  return (top == -1);  }  // 포화 상태 검출 함수  int is\_full()  {  return (top == (MAX\_STACK\_SIZE - 1));  }  void push(element item)  {  if (is\_full()) {  fprintf(stderr, "스택 포화 에러\n");  return;  }  else stack[++top] = item;  }  element pop()  {  if (is\_empty()) {  fprintf(stderr, "스택 공백 에러\n");  exit(1);  }  else return stack[top--];  }  element peek()  {  if (is\_empty()) {  fprintf(stderr, "스택 공백 에러\n");  exit(1);  }  else return stack[top];  }  int main(void)  {  element ie1 = { 20190001,  "Hong",  "Soeul" };  element ie2 = { 20190002,  "Hong1",  "Soeul1" };  element oe1;  element oe2;  push(ie1);  push(ie2);  oe1 = pop();  oe2 = pop();  printf("학번: %d\n", oe1.student\_no);  printf("이름: %s\n", oe1.name);  printf("주소: %s\n", oe1.address);  printf("학번: %d\n", oe2.student\_no);  printf("이름: %s\n", oe2.name);  printf("주소: %s\n", oe2.address);  return 0;  }  실행결과  -> LIFO 이므로 뒤늦게 들어온 ie2가 oe1에 정의되어 출력되었다. |
| 3. 프로그램 4.3  #include <stdio.h>  #include <stdlib.h>  #define MAX\_STACK\_SIZE 100  typedef int element;  typedef struct {  element data[MAX\_STACK\_SIZE];  int top;  } StackType;  void init\_stack(StackType\* s)  {  s->top = -1;  }  int is\_empty(StackType\* s)  {  return (s->top == -1);  }  int is\_full(StackType\* s)  {  return (s->top == (MAX\_STACK\_SIZE - 1));  }  void push(StackType\* s, element item)  {  if (is\_full(s)) {  fprintf(stderr, "스택 포화 에러\n");  return;  }  else s->data[++(s->top)] = item;  }  element pop(StackType\* s)  {  if (is\_empty(s)) {  fprintf(stderr, "스택 공백 에러\n");  exit(1);  }  else return s->data[(s->top)--];  }  element peek(StackType\* s)  {  if (is\_empty(s)) {  fprintf(stderr, "스택 공백 에러\n");  exit(1);  }  else return s->data[s->top];  }  int main(void)  {  StackType s;  init\_stack(&s);  push(&s, 1);  push(&s, 2);  push(&s, 3);  printf("%d\n", pop(&s));  printf("%d\n", pop(&s));  printf("%d\n", pop(&s));  }  실행결과  -> 구조체 s에 1 -> 2 -> 3을 push 하였으므로 pop의 결과는 3->2->1이 옳다 |
| 4. 프로그램 4.4  #include <stdio.h>  #include <stdlib.h>  #define MAX\_STACK\_SIZE 100  typedef int element;  typedef struct {  element data[MAX\_STACK\_SIZE];  int top;  } StackType;  void init\_stack(StackType\* s)  {  s->top = -1;  }  int is\_empty(StackType\* s)  {  return (s->top == -1);  }  int is\_full(StackType\* s)  {  return (s->top == (MAX\_STACK\_SIZE - 1));  }  void push(StackType\* s, element item)  {  if (is\_full(s)) {  fprintf(stderr, "스택 포화 에러\n");  return;  }  else s->data[++(s->top)] = item;  }  element pop(StackType\* s)  {  if (is\_empty(s)) {  fprintf(stderr, "스택 공백 에러\n");  exit(1);  }  else return s->data[(s->top)--];  }  element peek(StackType\* s)  {  if (is\_empty(s)) {  fprintf(stderr, "스택 공백 에러\n");  exit(1);  }  else return s->data[s->top];  }  int main(void)  {  StackType\* s;  s = (StackType\*)malloc(sizeof(StackType));  init\_stack(s);  push(s, 1);  push(s, 2);  push(s, 3);  printf("%d\n", pop(s));  printf("%d\n", pop(s));  printf("%d\n", pop(s));  free(s);  }  실행결과  -> 동적 메모리 할당을 이용해 스택을 구현하였으므로 결과값은 다른 스택과 동일 |
| 5. 프로그램 4.4 Quiz 답 설명  01 - 스택에서 top의 초기값은 무엇인가?   |  |  |  | | --- | --- | --- | |  | 메모리 할당 후 | init\_state(s); 후 | | TOP | 할당받은 address | -1 |   02 - 스택에서 top이 가리키는 것은 무엇인가?  void push(StackType\* s, element item)  {  if (is\_full(s)) {  fprintf(stderr, "스택 포화 에러\n");  return;  }  else s->data[++(s->top)] = item;  }  -> top은 push함수에서 스택이 full이 아닌 경우 stack에 item을 추가하며 top을 증가 시키도록 정의 되어있다. 결국, top은 -1부터 시작하여 item의 개수만큼 증가한다. top은 아이템의 위치를 가르킨다.   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | init | push(첫 번째 item) | push(두 번째 item) | push(세 번째 item) | | top | -1 | 0 | 1 | 2 |   -> top == stack.itemcount-1  03 - top이 8이면 현재 스택에 저장된 데이터의 총수는 몇 개인가?  -> 2번에서 top == stack.itemcount-1 임을 알았으니 top + 1 == stack.itemcount 임을 알 수 있다. 저장된 데이터의 총 수는 9개이다. |