|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』실습 보고서 | | | |
| 제목 | 4장 실습( O ) 과제( ) | 제출일자 | 2022.  10 .    12 . |
| 학번 | 201911608 | 이름 | 김지환 |

|  |
| --- |
| 1. 프로그램 4.6  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_STACK\_SIZE 100  typedef char element;  #define MAX\_STACK\_SIZE 100  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 check\_matching(const char\* in)  {  StackType s;  char ch, open\_ch;  int i, n = strlen(in);  init\_stack(&s);  for (i = 0; i < n; i++) {  ch = in[i];  switch (ch) {  case '(': case '[': case '{':  push(&s, ch);  break;  case ')': case ']': case '}':  if (is\_empty(&s)) return 0;  else {  open\_ch = pop(&s);  if ((open\_ch == '(' && ch != ')') ||  (open\_ch == '[' && ch != ']') ||  (open\_ch == '{' && ch != '}')) {  return 0;  }  break;  }  }  }  if (!is\_empty(&s)) return 0;  return 1;  }  int main(void)  {  char\* p = "{ A[(i+1)]=0; }";  if (check\_matching(p) == 1)  printf("%s 괄호검사성공\n", p);  else  printf("%s 괄호검사실패\n", p);  return 0;  }  실행 결과      ->  마지막 중괄호를 삭제 시 괄호검사실패가 출력되면서 올바르게 작성된 코드다. |
| 2. 프로그램 4.7  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_STACK\_SIZE 100  typedef char element;  #define MAX\_STACK\_SIZE 100  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 eval(char exp[])  {  int op1, op2, value, i = 0;  int len = strlen(exp);  char ch;  StackType s;  init\_stack(&s);  for (i = 0; i < len; i++) {  ch = exp[i];  if (ch != '+' && ch != '-' && ch != '\*' && ch != '/') {  value = ch - '0';  push(&s, value);  }  else {  op2 = pop(&s);  op1 = pop(&s);  switch (ch) {  case '+': push(&s, op1 + op2); break;  case '-': push(&s, op1 - op2); break;  case '\*': push(&s, op1 \* op2); break;  case '/': push(&s, op1 / op2); break;  }  }  }  return pop(&s);  }  int main(void)  {  int result;  printf("후위표기식은 82/3-32\*+\n");  result = eval("82/3-32\*+");  printf("결과값은 %d\n", result);  return 0;  }  실행결과    ->  후위표기식 82/3-32\*+ 를 중위 표기식으로 변환하면  (8/2-3)+3\*2 이다. 해당 값은 (4-3)+6 = 1+6 = 7이다.  옳은 결과 값이다. |
| 3. 프로그램 4.8  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_STACK\_SIZE 100  typedef char element;  #define MAX\_STACK\_SIZE 100  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 prec(char op) {  switch (op) {  case '(': case ')': return 0;  case '+': case '-': return 1;  case '\*': case '/': return 2;  }  return -1;  }  void infix\_to\_postfix(char exp[]) {  int i = 0;  char ch, top\_op;  int len = strlen(exp);  StackType s;  init\_stack(&s);  for (i = 0; i < len; i++) {  ch = exp[i];  switch (ch) {  case '+': case '-': case '\*': case '/':  while (!is\_empty(&s) && (prec(ch) <= prec(peek(&s))))  printf("%c", pop(&s));  push(&s, ch);  break;  case '(':  push(&s, ch);  break;  case ')':  top\_op = pop(&s);  while (top\_op != '(') {  printf("%c", top\_op);  top\_op = pop(&s);  }  break;  default:  printf("%c", ch);  break;  }  }  while (!is\_empty(&s))  printf("%c", pop(&s));  }  int main(void) {  char\* s = "(2+3)\*4+9";  printf("중위표시수식 %s \n", s);  printf("후위표시수식 ");  infix\_to\_postfix(s);  printf("\n");  return 0;  }  실행결과    ->  옳은 값인지 확인하기 위해 2번의 예시를 입력    옳다. |
| 4. 프로그램 4.9  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_STACK\_SIZE 100  #define MAZE\_SIZE 6  typedef struct { // 교체!  short r;  short c;  } 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];  }  element here = { 1,0 }, entry = { 1,0 };  char maze[MAZE\_SIZE][MAZE\_SIZE] = {  { '1', '1', '1', '1', '1', '1' },  { 'e', '0', '1', '0', '0', '1' },  { '1', '0', '0', '0', '1', '1' },  { '1', '0', '1', '0', '1', '1' },  { '1', '0', '1', '0', '0', 'x' },  { '1', '1', '1', '1', '1', '1' },  };  void push\_loc(StackType\* s, int r, int c)  {  if (r < 0 || c < 0) return;  if (maze[r][c] != '1' && maze[r][c] != '.') {  element tmp;  tmp.r = r;  tmp.c = c;  push(s, tmp);  }  }  void maze\_print(char maze[MAZE\_SIZE][MAZE\_SIZE])  {  printf("\n");  for (int r = 0; r < MAZE\_SIZE; r++) {  for (int c = 0; c < MAZE\_SIZE; c++) {  printf("%c", maze[r][c]);  }  printf("\n");  }  }  int main(void)  {  int r, c;  StackType s;  element x[100];  element y[100];  y[0].r = y[0].c = 0;  int idx = 1;  init\_stack(&s);  here = entry;  while (1) {  r = here.r;  c = here.c;  y[idx].r = r;  y[idx].c = c;  x[idx - 1].r = y[idx].r - y[idx - 1].r;  x[idx - 1].c = y[idx].c - y[idx - 1].c;  if (idx == 1) x[0].r = x[0].c = 0;  idx++;  if (maze[here.r][here.c] == 'x') break;  maze[r][c] = '.';  maze\_print(maze);  push\_loc(&s, r - 1, c);  push\_loc(&s, r + 1, c);  push\_loc(&s, r, c - 1);  push\_loc(&s, r, c + 1);  if (is\_empty(&s)) {  printf("실패\n");  return;  }  else  here = pop(&s);  }  printf("성공\n");  printf("\n ======path======\n");  for (int i = 1; i < idx;i++) {  printf(" {%d, %d} ", y[i].r, y[i].c);  printf("{%d, %d}\n", x[i-1].r, x[i-1].c);  }  return 0;  }  실행결과    -> 총 8단계이다.  경로의 출력 |
| 5. 프로그램 4.9 Quiz의 미로를 탐색하는 프로그램을 완성  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_STACK\_SIZE 100  #define MAZE\_SIZE 6  typedef struct { // 교체!  short r;  short c;  } 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];  }  element here = { 1,0 }, entry = { 1,0 };  char maze[MAZE\_SIZE][MAZE\_SIZE] = {  { '1', '1', '1', '1', '1', '1' },  { 'e', '0', '0', '0', '1', '1' },  { '1', '1', '0', '1', '1', '1' },  { '1', '1', '0', '1', '1', '1' },  { '1', '0', '0', '0', '0', 'x' },  { '1', '1', '1', '1', '1', '1' },  };  void push\_loc(StackType\* s, int r, int c)  {  if (r < 0 || c < 0) return;  if (maze[r][c] != '1' && maze[r][c] != '.') {  element tmp;  tmp.r = r;  tmp.c = c;  push(s, tmp);  }  }  void maze\_print(char maze[MAZE\_SIZE][MAZE\_SIZE])  {  printf("\n");  for (int r = 0; r < MAZE\_SIZE; r++) {  for (int c = 0; c < MAZE\_SIZE; c++) {  printf("%c", maze[r][c]);  }  printf("\n");  }  }  int main(void)  {  int r, c;  StackType s;  element x[100];  element y[100];  y[0].r = y[0].c = 0;  int idx = 1;  init\_stack(&s);  here = entry;  while (1) {  r = here.r;  c = here.c;  y[idx].r = r;  y[idx].c = c;  x[idx - 1].r = y[idx].r - y[idx - 1].r;  x[idx - 1].c = y[idx].c - y[idx - 1].c;  if (idx == 1) x[0].r = x[0].c = 0;  idx++;  if (maze[here.r][here.c] == 'x') break;  maze[r][c] = '.';  maze\_print(maze);  push\_loc(&s, r - 1, c);  push\_loc(&s, r + 1, c);  push\_loc(&s, r, c - 1);  push\_loc(&s, r, c + 1);  if (is\_empty(&s)) {  printf("실패\n");  return;  }  else  here = pop(&s);  }  printf("성공\n");  printf("\n ======path======\n");  for (int i = 1; i < idx;i++) {  printf(" {%d, %d} ", y[i].r, y[i].c);  printf("{%d, %d}\n", x[i-1].r, x[i-1].c);  }  return 0;  }  실행결과    -> 마지막 x 지점까지 합하면 총 10단계이다.  ---> 수정 사항.  char maze[MAZE\_SIZE][MAZE\_SIZE] 에서 미로를 수정  좌표 값을 가진 element 구조체를 이용한  element y[100]; --> index path  element x[100]; --> x, y path  y[idx] - y[idx-1] 로 x, y path를 획득  if 를 통한 while 탈출로 x이전 경로 까지 였지만 미로를 탐색했지만 이후 x까지의 경로를 획득. |