|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』실습 보고서 | | | |
| 제목 | 3장 실습( O ) 과제( ) | 제출일자 | 2022.   09.  29 . |
| 학번 | 201911608 | 이름 | 김지환 |

|  |
| --- |
| 실습11. 프로그램 3.7  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #define SIZE 6  void get\_integers(int list[]) {  printf("6개의 정수를 입력하시오: ");  for (int i = 0; i < SIZE; ++i)  scanf("%d", &list[i]);  }  int cal\_sum(int list[]) {  int sum = 0;  for (int i = 0; i < SIZE; ++i)  sum += \*(list + i);  return sum;  }  int main() {  int list[SIZE];  get\_integers(list);  printf("합 = %d \n", cal\_sum(list));  return 0;  }  실행결과    실습 12. 프로그램 3.7 Quiz 내용을 완성하여 실행 결과를 설명  01. Point가 2차원 공간에서의 점을 나타내는 구조체라고 했을 때 다음의 두 가지 함수 정의의 차이점은 무엇인까?  double get\_distance(Point p1, Point p2) { ... }  double get\_distance(Point \*p1, Point \*p2) { ... }  02. Point가 2차원 공간에서의 점을 나타내는 구조체라고 했을 때 다음의 두 가지 함수 정의의 차이점은 무엇인가? 어떤 경우에 포인터의 포언티러르 함수의 매개 변수로 전달하는가?  void sub1(Point \*p) { ... }  void sub2(Point \*\*p) { ... }  실행코드  #include <stdio.h>  typedef struct {  int x;  int y;  } Point;  double get\_distance1(Point p1, Point p2) {  printf("\nget\_distance1 --\n");  printf("p1.x : %d, p1.y : %d, p2.x : %d, p2.y :%d\n", p1.x, p1.y, p2.x, p2.y);  }  double get\_distance2(Point \*p1, Point \*p2) {  printf("\nget\_distance2 --\n");  printf("p1.x : %d, p1.y : %d, p2.x : %d, p2.y :%d\n", p1->x, p1->y, p2->x, p2->y);  }  void sub1(Point\* p) {  printf("\nsub1--\n");  printf("p.x : %d, p.y : %d \n", (p)->x, (p)->y);  }  void sub2(Point\*\* p) {  printf("\nsub2--\n");  printf("p.x : %d, p.y : %d \n", (\*p)->x, (\*p)->y);  }  int main() {  Point p = { 3,6 };  Point p1 = { 0, 0 };  Point p2 = { 6, 8 };  Point \*p3 = &p;  printf("main --\n");  printf("p.x = %d, p.y = %d\n", p.x, p.y);  printf("p1.x = %d, p1.y = %d\n", p1.x, p1.y);  printf("p2.x = %d, p2.y = %d\n", p2.x, p2.y);  get\_distance1(p1, p2);  get\_distance2(&p1, &p2);  sub1(&p);  sub2(&p3);  }  실행결과  문제 없이 잘 해결 되었다. |
| 실습 13. 프로그램 3.8  #include <stdio.h>  #include <stdlib.h>  #include <malloc.h>  #define SIZE 10  int main(void) {  int\* p;  p = (int\*)malloc(SIZE \* sizeof(int));  if (p == NULL) {  fprintf(stderr, "메모리가 부족해서 할당할 수 없습니다.\n");  exit(1);  }  for (int i = 0; i < SIZE; i++) p[i] = i;  for (int i = 0; i < SIZE; i++) printf("%d ", p[i]);  free(p);  return 0;  }  실행 결과    -> free(p) 이후 printf(”\n%d “, p[0]); 을 실행한 결과 알 수 없는 주소 값만 뜨는 것을 보아 동적으로 메모리 할당이 잘 되었음을 알 수 있다.  실습 14. 프로그램 3.9  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  typedef struct studentTag{  char name[10];  int age;  double gpa;  } student;  int main(void) {  student\* s;  s = (student\*)malloc(sizeof(student));  if (s == NULL) {  fprintf(stderr, "메모리가 부족해서 할당할 수 없습니다.\n");  exit(1);  }  strcpy(s->name, "Park");  s->age = 20;  s->gpa = 4.5;  printf("%s, %d, %2.1f", s->name, s->age, s->gpa);  free(s);  return 0 ;  }  실행 결과 |