|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2020\_1\_C\_9주 | 학번 : | 20175105 | 이름 : | 곽영주 |

* **강의 내용**

1. 함수 포인터 이용

|  |
| --- |
| [프로그램 소스]  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  void add(double \*, double, double);  void subtract(double \*, double, double);  int main(void) {  double m, n, result = 0;  void (\*pf)(double \*, double, double) = NULL; // 함수 포인터  printf("+, -를 수행 할 실수 2개를 입력하세요. >> ");  scanf("%lf %lf", &m, &n);  pf = add; // 사칙 연산을 수행  pf(&result, m, n); // add(&result, m, n);  printf("pf = %p, 함수 add() 주소 = %p\n", pf, add);  printf("더하기 수행: %.2lf + %.2lf = %.2lf\n\n", m, n, result);  pf = subtract;  pf(&result, m, n); // subtract(&result, m, n);  printf("pf = %p, 함수 add() 주소 = %p\n", pf, subtract);  printf("빼기 수행: %.2lf + %.2lf = %.2lf\n\n", m, n, result);    return 0;  }  void add(double \*z, double x, double y) {  \*z = x + y;  }  void subtract(double\* z, double x, double y) {  \*z = x - y;  } |
| [실행 결과] |

1. 정수와 실수를 모두 교환할 수 있는 프로그램

|  |
| --- |
| [프로그램 소스]  #include <stdio.h>  void exchange(char, void \*, void \*);  int main() {  int a = 10, b = 20;  double da = 1.5, db = 2.5;  exchange('c', &a, &b);  printf("정수 값 교환 후 : %d, %d\n", a, b);  exchange('d', &da, &db);  printf("실수 값 교환 후 : %.1lf, %.1lf\n", da, db);  return 0;  }  void exchange(char ch, void \*vp1, void \*vp2) { // type은 자료형을 구분하기 위한 매개변수  int itp;  double dtp;  switch (ch) {  case 'c':  itp = \*(int \*)vp1;  \*(int \*)vp1 = \*(int \*)vp2;  \*(int \*)vp2 = itp;  break;  case 'd':  dtp = \*(double\*)vp1;  \*(double\*)vp1 = \*(double\*)vp2;  \*(double\*)vp2 = dtp;  break;  default:  printf("해당 사항 없음\n");  }  } |
| [실행 결과] |

1. 매개변수로 포인터 전달

|  |
| --- |
| [프로그램 소스]  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  void func(int (\*)(int, int));  int sum(int, int);  int mul(int, int);  int main() {  int sel;    printf("1. 두 정수의 합\n");  printf("2. 두 정수의 곱\n");  printf("원하는 작업을 선택하세요 : ");  scanf("%d", &sel);  switch (sel) {  case 1:  func(sum);  break;  case 2:  func(mul);  break;  }  return 0;  }  void func(int (\*fp)(int, int)) {  int a, b;  int res;  printf("두 정수 값을 입력하세요 : ");  scanf("%d %d", &a, &b);  res = fp(a, b);  printf("결과값은 : %d\n", res);  }  int sum(int a, int b) {  return a + b;  }  int mul(int a, int b) {  return a \* b;  } |
| [실행 결과] |

1. 전역 변수와 지역 변수의 선언과 참조

|  |
| --- |
| [프로그램 소스]  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  void infunction(void);  void outfunction(void);  int global = 10; // 전역 변수  static int sglobal = 20; // 정적 전역 변수  int main(void) {  int x = 100; // main() 함수의 지역 변수    printf("%d, %d, %d\n", global, sglobal, x);  infunction();  outfunction();  infunction();  outfunction();  infunction();  outfunction();  printf("global: %d, sglobal: %d, x: %d\n", global, sglobal, x);  return 0;  }  void infunction(void) {  auto int fa = 1; // infunction() 함수의 자동 지역 변수  static int fs; // infunction() 함수의 정적 지역 변수  printf("global: %d, sglobal: %d, fa: %d, fs: %d\n", ++global, ++sglobal, fa, ++fs);  }  -----------------------------------------------------------------------------------------------------------  #include <stdio.h>  void outfunction() {  extern int global;  printf("%d\n", ++global);  } |
| [실행 결과] |

* **내용 점검**

1. 다음에서 서술 내용이 맞으면 0, 틀리면 x 하시오.
2. 지역변수는 함수 또는 블록에서 선언된 변수이다. ( O )
3. 지역변수는 선언 후 초기화하지 않으면 0과 같은 기본 값이 저장된다. ( X )
4. 전역변수는 선언되면 자동으로 초기 값이 자료형에 맞는 0으로 지정된다. ( O )
5. 함수나 블록에서 전역변수와 이름이 같은 지역변수를 선언할 수 있다. 이런 경우, 함수내부나 블록에서 그 이름을 참조하면 전역변수로 인식한다. ( X )
6. 레지스터 변수는 일반 변수와 같이 주소연산자 &를 사용할 수 있다. ( X )
7. 정적변수는 초기 값을 지정하지 않으면 자동으로 자료형에 따라 0이나 ‘\0’ 또는 NULL 값이 저장된다. ( O )
8. C 언어에서 포인터를 매개변수로 사용하면 함수로 전달된 실인자의 주소를 이용하여 그 변수를 참조할 수 있는 특징이 있다. 이와 같이 함수에서 주소의 호출을 참조에 의한 호출이라 한다. ( O )
9. 전역변수의 사용은 모든 함수에서 공유할 수 있는 저장공간을 이용할 수 있는 장점이 있으나, 어느 한 함수에서 잘못 다루면 모든 함수에 영향을 미치는 단점도 있다. ( O )
10. 다음에서 비어있는 부분을 적당히 채우시오.
11. 변수의 참조가 유효한 범위를 변수의 ( 유효범위 )라 한다.
12. 키워드 ( extern )을 사용한 변수 선언은 새로운 변수를 선언하는 것이 아니라 단지 이미 존재하는 전역변수의 유효 범위를 확장한다.
13. ( 정적 전역변수 )는 선언된 파일 내부에서만 참조가 가능한 변수이다.
14. 메인 메모리의 영역은 프로그램 실행 과정에서 ( 데이터(data) ) 영역, ( 힙(heap) ) 영역, ( 스택(stack) ) 영역 세 부분으로 나뉜다.
15. ( 힙(heap) )영역은 동적 할당(dynamic allocation)되는 변수가 할당되는 저장공간이다.
16. 해당 파일 내부에서만 변수를 공유하려면 ( 정적 전역변수 )를 이용한다.
17. ( 정적 지역변수 ) 변수는 함수나 블록을 종료해도 메모리에서 제거되지 않고 계속 메모리에 유지 관리되는 특성이 있다.

* **프로그래밍 과제**

1. 제시된 조건에 따라 함수를 작성한 후 함수 포인터를 이용하여 호출하도록 프로그램 하시오

* 직원들의 월급이 배열A에 저장되어 있다고 가정하여 회사에서 지급할 월급의 총액을 계산하여 반환하는 함수

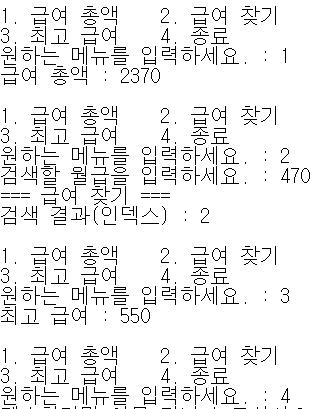
int array\_sum(int \*A, int size)

* 직원들의 월급이 저장된 배열에서 입력된 데이터와 일치하는 급여를 가진 배열의 인덱스를 반환하는 함수, 데이터 입력은 함수에서 받는다

int array\_search(int \*A, int size)

* 직원들의 월급이 저장된 배열에서 가장 많은 급여를 반환하는 함수

int array\_max(int \*A, int size)

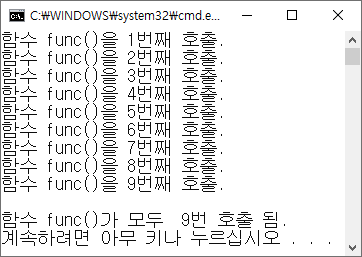
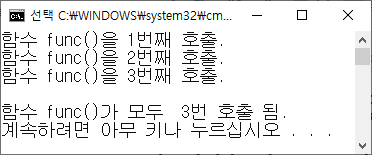


|  |
| --- |
| **[프로그램 소스]**  #include <stdio.h>  int array\_sum(int \*, int);  int array\_search(int \*, int);  int array\_max(int \*, int );  int main() {  int A[] = { 200, 300, 220, 410, 180 }; // 급여  int size = sizeof(A) / sizeof(A[0]); // 배열크기  int (\*fp)(int \*, int) = NULL; // 함수 포인터  int menu; // 메뉴  do {  printf("1. 급여총액\t 2. 급여찾기\n");  printf("3. 최고급여\t 4. 종료\n");  printf("원하는 메뉴를 입력하세요: ");  scanf\_s(" %d", &menu);  switch (menu) {  case 1: // 급여 총액  fp = array\_sum;  printf("급여총액 : %d\n", fp(A, size));  puts("");  break;  case 2: // 급여 찾기  fp = array\_search;  printf("검색 결과(인덱스) : %d\n", fp(A, size));  puts("");  break;  case 3: // 최고급여  fp = array\_max;  printf("최고급여 : %d\n", fp(A, size));  puts("");  break;  case 4: // 종료  printf("종료\n");  break;  }  } while (menu != 4);  return 0;  }  int array\_sum(int\* A, int size) { // 급여 총액  int i, sum = 0;    for (i = 0; i < size; i++) {  sum += \*(A + i);  }  return sum;  }  int array\_search(int\* A, int size) { // 급여 찾기  int i, search\_salary, index = 0;  printf("검색할 월급을 입력하세요 : ");  scanf\_s(" %d", &search\_salary);  printf("=== 급여 찾기 ===\n");    for (i = 0; i < size; i++) {  if (search\_salary == \*(A + i)) {// 검색한 급여를 찾은 경우  index = i;  break;  }  else // 못 찾은 경우 -1  index = -1;  }  return index;  }  int array\_max(int\* A, int size) { // 최고 급여  int i, max = A[0]; // max값을 배열 첫번째 값으로 초기화  for (i = 1; i < size; i++) {  if (max >= \*(A + i)) // max에 저장 된 값이 크거나 같은 경우  continue;  else // max에 저장된 값이 작은 경우  max = \*(A + i);  }  return max;  } |
| **[실행 결과]** |

1. 1번 문제를 함수 포인터 배열을 이용하여 호출하도록 프로그램 하시오.

|  |
| --- |
| **[프로그램 소스]**  #include <stdio.h>  int array\_sum(int \*, int);  int array\_search(int \*, int);  int array\_max(int \*, int );  int main() {  int A[] = { 200, 300, 220, 410, 180 }; // 급여  int size = sizeof(A) / sizeof(A[0]); // 배열크기  int (\*fp[3])(int\*, int) = { array\_sum, array\_search, array\_max }; // 함수 포인터 배열  int menu; // 메뉴  do {  printf("1. 급여총액\t 2. 급여찾기\n");  printf("3. 최고급여\t 4. 종료\n");  printf("원하는 메뉴를 입력하세요: ");  scanf\_s(" %d", &menu);  switch (menu) {  case 1: // 급여 총액  printf("급여총액 : %d\n", fp[0](A, size));  puts("");  break;  case 2: // 급여 찾기  printf("검색 결과(인덱스) : %d\n", fp[1](A, size));  puts("");  break;  case 3: // 최고급여  printf("최고급여 : %d\n", fp[2](A, size));  puts("");  break;  case 4: // 종료  printf("종료\n");  break;  }  } while (menu != 4);  return 0;  }  int array\_sum(int\* A, int size) { // 급여 총액  int i, sum = 0;    for (i = 0; i < size; i++) {  sum += \*(A + i);  }  return sum;  }  int array\_search(int\* A, int size) { // 급여 찾기  int i, search\_salary, index = 0;  printf("검색할 월급을 입력하세요 : ");  scanf\_s(" %d", &search\_salary);  printf("=== 급여 찾기 ===\n");    for (i = 0; i < size; i++) {  if (search\_salary == \*(A + i)) {// 검색한 급여를 찾은 경우  index = i;  break;  }  else // 못 찾은 경우 -1  index = -1;  }  return index;  }  int array\_max(int\* A, int size) { // 최고 급여  int i, max = A[0]; // max값을 배열 첫번째 값으로 초기화  for (i = 1; i < size; i++) {  if (max >= \*(A + i)) // max에 저장 된 값이 크거나 같은 경우  continue;  else // max에 저장된 값이 작은 경우  max = \*(A + i);  }  return max;  } |
| **[실행 결과]** |

1. 함수 자신이 몇 번 호출 되었는지를 main() 함수로 반환하여 출력하는 프로그램을 작성하시오. 단, 전역 변수를 사용하지 않고 정적 변수를 사용하며 프로그램을 실행할 때 마다 함수 호출 횟수는 제시된 결과처럼 서로 다르다

|  |
| --- |
| **[프로그램 소스]**  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  void func();  int main() {  srand((long)time(NULL));  int i, ran = rand() % 10 + 1;  for (i = 0; i < ran; i++) {  func();  }  puts("");  printf("함수 func()가 모두 %d번 호출 됨.\n", ran);  return 0;  }  void func() {  static int n = 0; // 정적 변수  printf("함수 func()를 %2d번째 호출.\n", ++n);  } |
| **[실행 결과]** |