|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C2\_실습과제\_5주** | **이름 :** | **이갑성** | **학번 :** | **20135151** |

* **강의내용**

1. 포인터 변수의 값과 주소의 표현

int var=30;

int \*pt=&var;

printf(“pt : %p”, pt); //포인터 변수에 저장된 주소 출력

printf(“\*pt : %d”, pt); //포인터가 가리키는 변수 var값 출력(간접참조)

1. 포인터 배열

- 배열의 원소가 주소 값을 저장한 포인터

int a, b, c;

int \*pary[3] = {&a, &b, &c};

1. 배열 포인터

* 포인터에 배열의 시작 주소를 저장

int a[] = {3,2,1};

int \*pa=a; **//일차원 배열 포인터, \*(pa+1)? pa[1]? 둘 다 가능하다.**

int b[][3] = {1,2,3,4,5,6};

int (\*pb)[3]= b; **//이차원 배열 포인터, \*(b[1]+1)? \*(\*(pb+1)+1)? pb[1][1]? 셋 다 가능하다.**

1. 함수 간 통신을 위한 포인터

-함수에서의 배열 전달 : 주소 전달

-포인터를 이용한 함수 간 데이터 접근

-포인터와 배열의 혼용

1. 함수 포인터 / 함수 포인터 배열

* 함수 포인터

void afunc(int \*, int, int);

void bfunc(int \*, int, int);

void (\*pf)(int \*, int, int) = null;

//또는 함수 포인터 선언과 동시에 초기화 void (\*pf)(int \*, int, int) = afunc;

pf=afunc; //함수 포인터 초기화

pf(&result, m, n); //함수 포인터를 사용하여 함수 호출

* 함수 포인터 배열

void (\*pfary[2])(int \*, int, int) = null;

pfary[0] = afunc;

pfary[1] = bfunc;

void (\*pfary[2])(int \*, int, int) = {afunc, bfunc}; //함수 포인터 배열 선언과 동시에 초기화

pfary[0](&result, m, n);

pfary[1](&result, m, n);

* **내용 점검**

1. 다음 설명에 대한 변수 선언문을 작성하시오.
2. 함수원형이 void functionname(int, int)인 함수 포인터 pf :

=> void (\*pf)(int, int);

함수원형이 int \*functionname(double, double)인 함수 포인터 addf : //반환 형태가 int 포인터

=> int \*(\*addf)(double, double);

1. 함수원형이 double functionname(double \*, double)인 함수 포인터 3개인 배열 aryfp :

=> void (\*aryfp[3])(double \*, double)

1. 다음 문장에서 문법오류 및 실행오류를 찾아 수정하시오
2. double data[2][3]; double (\*pd)[2] = data;

=> double (\*pd)[3] = data;

1. int add(int, int); int (\*pf)(int, int) = &add;

=> int (\*pf)(int, int) = add;

1. int add(int, int); int sub(int, int);

int (\*pf[2]) = {add, sub};

=> int (\*pf[2])(int, int) = {add, sub};

* 프로그램 과제 1

**다음 일차원 배열과 포인터 선언 문장에 대하여 다음 각각의 연산식 결과를 기술하시오.**

int score[][3] = {3, 5, 6, 9, 2, 5, 7, 8, 9, 1};

int (\*ps)[3] = score;

int \*p = score[0];

int \*q = score[1];

1. \*ps

주소값

1. ps[2][2]

9

1. \*\*score

3

1. \*p

3

1. \*q

9

1. \*score[2]

7

1. \*(score[2]+1)

8

1. \*(\*(score+1)+1)

2

1. \*(\*score+5)

2

1. (\*(score+3))[2]

0

**2. 자료유형 double형 1차원 배열에서 가장 큰 값과 가장 작은 값을 찾아 그 값의 차이를 반환하는 함수를 만들어 결과를 알아보는 프로그램을 작성하시오.**

#include <stdio.h>

#define SIZE 5

void print(double \*x, int size);

double diffminmax(double \*a, int size);

int main(void)

{

double source[5] = {3.12, 5.14, 7.25, 7.48, 5.91};

printf("source 원배열값: \n");

print(source, SIZE);

printf("\n 최대값과 최소값의 차이는 %.2lf 이다. \n", diffminmax(source, SIZE));

return 0;

}

|  |
| --- |
| * 프로그램 소스   #include <stdio.h>  #define SIZE 5  void print(double \*x, int size);  double diffminmax(double \*a, int size);  int main(void)  {  double source[5] = { 3.12, 5.14, 7.25, 7.48, 5.91 };  printf("source 원배열값: \n");  print(source, SIZE);  printf("\n 최대값과 최소값의 차이는 %.2lf 이다. \n", diffminmax(source, SIZE));  return 0;  }  void print(double \*x, int size)  {  int i;  for (i = 0; i < size; i++)  {  printf("%.2lf ", \*(x + i));  }  }  double diffminmax(double \*a, int size)  {  double max = 0.0;  double min = 100.0;  int i;  for (i = 0; i < size; i++)  {  if (max < \*(a + i)) // 최대값 찾기  max = \*(a + i);    if (min > \*(a + i)) // 최소값 찾기  min = \*(a + i);  }  return max - min;  } |
| * 실행결과 |

**3. 다음을 참고로 [파스칼의 삼각형]을 생성하고 출력하는 프로그램을 작성하시오.**

* 정수형 배열 pascal[10][10]을 선언하여 10행까지 [파스칼의 삼각형]을 저장 후 출력
* size 크기의 파스칼 삼각형을 이차원 배열 p에 생성하는 함수

void pascaltriangle(int (\*p)[10], int size);

* 하나의 행에서 처음과 끝이 아니면 p[i][j] = p[i-1][j] + p[i-1][j-1];
* 하나의 행에서 처음 또는 끝이면 p[i][j] = 1;
* size 크기 배열 p를 그림과 같이 삼각형만 출력하는 함수

void triangleprint(int (\*p)[10], int size)

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

#include <stdio.h>

#define ROW 10

void pascaltriangle(int (\*p)[ROW], int size); //파스칼 삼각형 만들기

void triangleprint(int (\*p)[ROW], int size); //파스칼 삼각형 출력

int main(void)

{

int pascal[10][10] = {0};

pascaltriangle(pascal, ROW);

triangleprint(pascal, ROW);

return 0;

}

|  |
| --- |
| * 프로그램 소스   #include <stdio.h>  #define ROW 10  void pascaltriangle(int(\*p)[ROW], int size); //파스칼 삼각형 만들기  void triangleprint(int(\*p)[ROW], int size); //파스칼 삼각형 출력  int main(void)  {  int pascal[10][10] = { 0 };  pascaltriangle(pascal, ROW);  triangleprint(pascal, ROW);  return 0;  }  void pascaltriangle(int(\*p)[ROW], int size)  {  int i, j;  for (i = 0; i < size; i++)  {  for (j = 0; j <= i; j++)  {  if ((i == j) ||(j == 0))  {  p[i][j] = 1;  }  else  {  p[i][j] = p[i - 1][j] + p[i - 1][j - 1];  }  }  }  }  void triangleprint(int(\*p)[ROW], int size)  {  int i, j;    for (i = 0; i < size; i++)  {  for (j = 0; j <= i; j++)  {  printf("%d ", \*(\*(p + i) + j));    }  printf("\n");  }       * } |
| * 실행결과 |

**4. 다음을 참고로 두 일차원 배열의 병합을 구하여 출력하는 프로그램을 작성하시오.**

- 이미 오름차순으로 정렬되어 있는 두 배열 a, b의 병합 결과 c는 다음과 같음

int a[] = {1, 2, 5, 7, 9, 14};

int b[] = {2, 3, 6, 8, 13};

int c[] = {1, 2, 2, 3, 5, 6, 7, 8, 9, 13, 14};

- 배열 a, b의 병합 결과를 배열c에 저장하는 함수

(an=a배열크기, bn= b배열크기)

void arraymerge(int \*a, int an, int \*b, int bn, int \*c)

#include <stdio.h>

void arraymerge(int \*a, int an, int \*b, int bn, int \*c);

void print(int \*p, int size);

void main(void)

{

int a[] = {1, 2, 5, 7, 9, 14};

int an = sizeof(a)/sizeof(a[0]);

int b[] = {2, 3, 6, 8, 13};

int bn = sizeof(b)/sizeof(b[0]);

int c[20] = {0};

arraymerge(a, an, b, bn, c);

print(c, an+bn);

}

|  |
| --- |
| 프로그램소스  #include <stdio.h>  void arraymerge(int \*a, int an, int \*b, int bn, int \*c);  void print(int \*p, int size);  void main(void)  {  int a[] = { 1, 2, 5, 7, 9, 14 };  int an = sizeof(a) / sizeof(a[0]);  int b[] = { 2, 3, 6, 8, 13 };  int bn = sizeof(b) / sizeof(b[0]);  int c[20] = { 0 };  arraymerge(a, an, b, bn, c);  print(c, an + bn);  }  void arraymerge(int \*a, int an, int \*b, int bn, int \*c)  {  int i, j, c\_size, k, z, min, temp;  for (i = 0; i < an; i++)  {  \*(c + i) = \*(a + i);  }    for (j = 0; j < bn; j++)  {  \*((c +i) + j) = \*(b + j);  }  c\_size = i + j;    for (k = 0; k < c\_size - 1; k++)  {  min = k;  for (z = k + 1; z < c\_size; z++)  {  if (\*(c + z) < \*(c + min))  {  min = z;  }  }  temp = \*(c + k);  \*(c + k) = \*(c + min);  \*(c + min) = temp;    }  }  void print(int \*p, int size)  {  int i;  for (i = 0; i < size; i++)  {  printf("%d ", \*(p + i));  }  } |
| 결과 |

**5. 다음을 참고로 표준입력으로 받은 두 실수의 사칙연산을 수행하는 프로그램을 작성하시오.**

- 사칙연산을 수행하는 함수를 모두 4개 정의하여 이를 함수 포인터 배열에 저장하여 수행

- 사칙연산 중에서 하나의 연산을 다음과 같이 표준입력으로 받음

사칙연산을 위하여 각 연산에 대한 번호를 입력하세요.

[더하기]: 0, [빼기]: 1, [곱하기]: 2, [나누기]: 3 >> 0

사칙연산을 수행할 실수 2개를 입력하세요. >> 22.3 33.4

더하기 수행 : 22.300000 + 33.400000 = 55.700000

#include <stdio.h>

void add(double \*z, double x, double y) { \*z = x + y; }

void subtract(double \*z, double x, double y) { \*z = x - y; }

void multiply(double \*z, double x, double y) { \*z = x \* y; }

void devide(double \*z, double x, double y) { \*z = x / y; }

|  |
| --- |
| * 프로그램 소스   #include <stdio.h>  void add(double \*z, double x, double y);  void subtract(double \*z, double x, double y);  void multiply(double \*z, double x, double y);  void devide(double \*z, double x, double y);  void main()  {  int select;  double n1, n2, result;  void(\*pAry[4])(double \*, double, double) = { add, subtract, multiply, devide };  printf("사칙연산을 위하여 각 연산에 대한 번호를 입력하세요. \n");  printf("[더하기]: 0, [빼기]: 1, [곱하기]: 2, [나누기]: 3 >> \n");  scanf("%d", &select);  printf("사칙연산을 수행할 실수 2개를 입력하세요. >> \n");  scanf("%lf %lf", &n1, &n2);  if (select == 0)  {  pAry[select](&result, n1, n2);  printf("%lf + %lf = %lf", n1, n2, result);  }  else if (select == 1)  {  pAry[select](&result, n1, n2);  printf("%lf - %lf = %lf", n1, n2, result);  }  else if (select == 2)  {  pAry[select](&result, n1, n2);  printf("%lf \* %lf = %lf", n1, n2, result);  }  else if (select == 3)  {  pAry[select](&result, n1, n2);  printf("%lf / %lf = %lf", n1, n2, result);  }  }  void add(double \*z, double x, double y)  {  \*z = x + y;  }  void subtract(double \*z, double x, double y)  {  \*z = x - y;  }  void multiply(double \*z, double x, double y)  {  \*z = x \* y;  }  void devide(double \*z, double x, double y)  {  \*z = x / y;  } |
| * 실행결과 |