|  |
| --- |
| 2020년 1학기 |
| 프로그래밍 언어 레포트 #11 |
| 12주차 학습활동 |

|  |
| --- |
| 송 지민  사회학과 201721899 |

**[과제#1] 참조에 의한 호출을 이용한 난수 생성**

* **코드 및 실행화면**

#include <stdio.h> // 표준입출력 헤더파일 선언

#include <stdlib.h> // 표준 라이브러리 선언(난수 생성)

// 함수 원형 정의

void array\_fill(int\* A, int size); // 난수를 받아서 저장하는 함수

void array\_printf(int\* A, int size);// 받은 난수 출력하는 함수

/\* 일반적으로 메인 함수가 우선하므로 메인 함수에서 쓰이는 함수는 변수를 선언하는 것처럼 미리 원형을 정의해야 한다.

원형을 정의할 때는 함수 선언 시와 마찬가지로 반환형, 함수 이름, 매개변수를 포함하여 쓰고 세미콜론을 붙여준다.

위의 함수에서 매개변수로 포인터가 사용되었다. 포인터란 메모리의 주소를 가지고 있는 변수를 말한다.

메모리에는 바이트 단위로 주소가 매겨져 있으며 자료형마다 차지하는 메모리의 크기가 달라지므로 이를 유의하여야 한다.

(char 1, int 4, long 4, long long 8, float 4, double 8)\*/

int main(void) // 반환값이 정수형이며 매개변수가 없는(void) 메인 함수의 선언부

{

// 배열 선언

int data[10]; // 크기가 10인 정수형 1차원 배열

// 함수 호출

array\_fill(data, 10); // 난수를 받아서 배열에 저장하는 함수 호출

array\_printf(data, 10); // 난수를 출력하는 함수 호출

/\* 함수 호출 시에는 반환형 없이 이름과 매개변수(인수)로 호출하며 함수 선언 시 매개변수를 정수형으로 선언했으면 인수 또한 정수가 되어야 한다.

일반적으로 반환값이 있는 함수이면 반환값을 메인 함수의 변수에 저장하여 사용하는 것으로 보이며, 반환형과 매개변수가 없을 수도 있다.

매개변수가 포인터 변수인 함수에 인수를 전달할 때 일반 변수의 경우에는 주소 연산자를 포함하여 그 위치를 전달하지만

여기서는 인수가 배열이므로 주소 연산자를 포함하지 않는 것으로 보인다.\*/

// 반환값

return 0; // 0 반환하여 메인 함수 끝내기

}

void array\_fill(int\* A, int size) // 반환값이 없으며 매개변수가 포인터, 정수형 변수인 함수

{

int i; // 인덱스 변수 선언

for (i = 0; i < size; i++) // 0부터 size-1까지, size만큼 반복

{

A[i] = rand(); // 배열 표기법을 사용하여 배열에 접근, 난수 저장

/\* 포인터와 배열을 아주 밀접한 관계를 가지고 있으며 배열 이름이 곧 포인터를 의미한다.

따라서 포인터를 배열처럼 사용이 가능하므로, 함수에서 포인터 매개변수를 사용할 때도 배열 매개변수와 같은 방식으로 사용할 수 있다.

(포인터 매개변수 사용 시 배열처럼 생각하고 코드를 작성하면 혼동을 줄일 수 있을 것으로 보임)

만약 배열의 이름을 인덱스 없이 출력하면 배열 시작점의 주소가 출력되며 배열이름 +1 을 출력하면 자료형에 따라 1(char형), 4(int)형 등이 더해져서 출력된다.

또한 \*배열이름 과 같은 형태로 출력하면 포인터를 사용할 때와 유사하게 배열 시작 주소에 저장되어 있는 값이 출력된다.\*/

}

}

void array\_printf(int\* A, int size) // 반환값이 없으며 포인터, 정수형 매개변수를 갖는 함수

{

int i; // 인덱스 변수 선언

for (i = 0; i < size; i++) // 0부터 size-1까지, size만큼 반복

{

printf("%d ", A[i]); // 반복문을 사용하여 배열을 출력

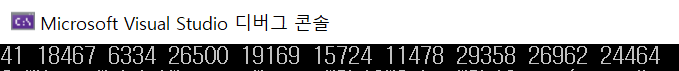
/\* 마찬가지로 포인터 매개변수를 사용했으므로 배열을 사용할 때처럼 인덱스를 주어서 반복문을 통해 차례대로 출력할 수 있다.

여기서는 배열에 저장되어 있는 값을 출력하기 때문에 형식 지정자로 %d를 사용하지만 주소를 출력하는 형식 지정자는 %p라는 점을 유의한다.

%p는 주소를 16진수로 출력하나 알기 쉽게 출력하기 위해 %u, 즉 부호가 없는 10진 정수 형식으로 출력하기도 한다.\*/

}

}



* **int 형과 long long 형을 이용한 주소 바이트 증가 실험**

(long long의 크기가 헷갈려서 long long을 활용하였습니다.)

#include <stdio.h>

int main(void)

{

// 변수 선언 및 초기화

int i = 0;

long long k = 0;

// 포인터 선언 및 초기화

int \*pi = &i;

long long \*pk = &k;

printf("증가 전\n");

printf("i의 주소 : %u\n", pi);

printf("k의 주소 : %u\n\n", pk);

pi++;

pk++;

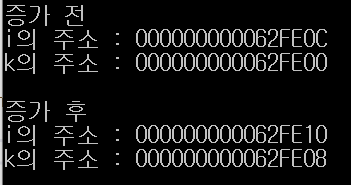
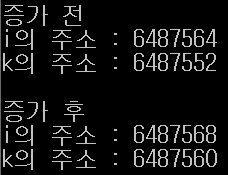
printf("증가 후\n");

printf("i의 주소 : %u\n", pi);

printf("k의 주소 : %u\n", pk);

return 0;

}



형식 지정자를 %u로 하여 출력한 화면

형식 지정자를

%p로 하여 출력한 화면

**[과제#2] 배열 요소 출력 함수**

* **코드 및 실행화면**

#include <stdio.h> // 표준 입출력 헤더파일 선언

void array\_printf(int\* A, int n); // 반환값이 없으며 포인터, 정수형 매개변수를 갖는 함수 원형 정의

// 메인 함수에서 쓰이는 함수가 메인 함수의 뒤에 오므로 함수 원형을 먼저 정의해 주어야 함

int main(void) // 반환값이 정수형이며 매개변수가 없는 메인 함수의 선언부

{

int list[10] = { 1, 2, 3, 4 }; // 크기가 10인 정수형 배열 선언 및 초기화

/\* 일차원 배열의 경우 선언과 동시에 초기화를 하면 배열의 크기를 명시하지 않아도 초기화된 값의 개수대로 배열의 크기가 정해진다.

이차원 배열의 경우 마찬가지로 행의 크기는 명시하지 않아도 되지만 열의 크기는 반드시 명시해야 함을 유의한다.

또한 배열을 따로 초기화하지 않으면 기본값으로 0이 저장되어 있다.

위 코드의 경우 10개 중 4개만 초기화했으므로 0부터 3까지 차례대로 값이 저장되며 4부터 9까지는 0이 저장되어 있다.\*/

array\_printf(list, 10); // 배열을 출력하는 함수를 호출, 인수로 배열과 배열의 크기를 전달

/\* 포인터 매개변수를 갖는 함수에 배열을 전달할 때는 일반 변수를 전달할 때와 다르게 주소 연산자를 포함하지 않아도 되는데,

이는 실질적으로 배열이 포인터와 유사한 개념이기 때문으로 보인다.

또한 배열을 전달할 때는 배열의 자료형과 주소를 제외하고 이름만 전달한다.\*/

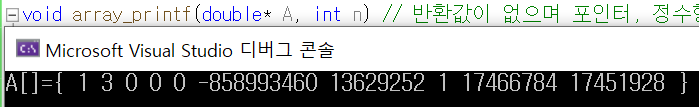
return 0; // 메인 함수에 0 반환(반환형이 정수이므로)하여 끝내기

}

void array\_printf(double\* A, int n) // 반환값이 없으며 포인터, 정수형 매개변수를 갖는 함수 선언부

/\* 주소는 한 가지 형식인데 가리키는 대상에 따라 타입을 다르게 하는 이유는 포인터가 가리키는 대상을 확실하게 함으로써 프로그래밍 실수를 예방하기 위해서이다.

포인터 타입을 다르게 해도 출력은 되지만 원래 값과 다른 값이 나오므로(실행화면 첨부)

가리키는 대상에 맞게 포인터 타입을 설정하는 것에 유의해야 한다.\*/

{

int i; // 인덱스 변수 선언

printf("A[]={ "); // 배열 출력 안내

// 반복문 이용하여 배열값 출력하기

for (i = 0; i < n; i++) // 0부터 배열의 크기만큼 반복하기

{

printf("%d ", A[i]);// 포인터 매개변수를 배열처럼 이용하여 출력하기

/\* 배열과 포인터는 유사한 개념이며 코드를 작성함에 있어서도 유사한 기능을 하도록 사용할 수 있다.

따라서 매개변수가 포인터일 때는 배열을 사용할 때와 유사하게 인덱스를 주어 출력이 가능하다.\*/

}

printf("}\n"); // 배열 출력 마무리

}

**[과제#3] 배열을 이용한 급여 계산 함수**

* **코드 및 실행화면**

#include <stdio.h> // 표준입출력 헤더파일 선언

/\* 포인터가 아무것도 가리키고 있지 않을 때는 NULL로 초기화해야 하는데 이 때 NULL은 stdio.h에 0으로 저장되어 있다(정확히는 #define NULL((void\*)0)).\*/

#define N\_DATA 10 // 기호상수 선언(배열의 크기)

void array\_add(int a[], int b[], int c[], int size);

// 반환값이 없으며 매개변수로 배열 3개와 정수형(배열의 크기) 변수를 받는 함수의 원형 정의

void array\_printf(char\* name, int\* a, int size)

// 반환값이 없으며 매개변수로 포인터, 정수형 변수를 받는 함수의 선언부

/\* 포인터가 가리키는 주소에 저장된 내용을 읽어오기 위해서는 포인터명 앞에 \*기호를 붙인다(\*p).

\*p는 자료형에 따라 그에 맞는 바이트를 읽어들이는데 예를 들어 int형 포인터이면 p가 가리키는 위치에 정수가 있다고 생각하고 4바이트를 읽어들인다.\*/

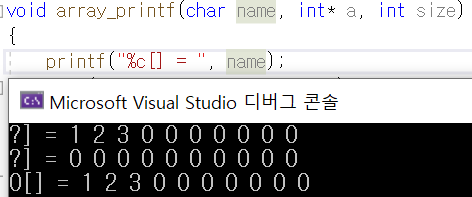
{

printf("%c[] = ", \*name); // 배열의 이름을 출력

/\* 배열 이름 출력 방식 : 문자형으로 배열의 이름을 받아 그 배열 이름이 가리키는 위치로 가서 배열명을 꺼내오는 것으로 보인다.

배열의 이름이 곧 포인터이기 때문에 "" 안에 배열의 이름을 주어 포인터로서 기능하도록 작성하였으며 포인터를 통해 원본 배열을 변경할 수 있다.

간접 참조 연산자(\*연산자)를 지우고 실행하였더니 오류가 발생하는 것을 확인하였다.\*/



// 배열 출력하는 반복문

for (int i = 0; i < size; i++) // 0부터 size만큼 반복

{

printf("%d ", a[i]); // 반복문을 통해 배열에 인덱스를 주어 출력

/\* 배열은 포인터와 유사하기 때문에 포인터 매개변수를 배열과 같은 형태로 사용할 수 있다.\*/

}

printf("\n"); // 줄바꿈

}

int main(void) // 반환값이 정수형이며 매개변수가 없는 메인 함수의 선언부

{

// 일차원 배열 선언 및 초기화

int A[N\_DATA] = { 1, 2, 3 }; // 기본급 배열 선언 및 초기화

int B[N\_DATA] = { 0 }; // 보너스 배열 선언 및 초기화

int C[N\_DATA] = { 0 }; // 월급의 총액 배열 선언 및 초기화

/\* 배열 크기는 모두 기호상수(10)이며 A같은 경우 0, 1, 2 인덱스에 해당되는 값만 초기화하였으므로 나머지 값은 0이 저장되어 있다.\*/

array\_printf("A", A, N\_DATA); // 배열 출력하는 함수에 첫번째 배열이름, 배열, 배열크기 인수로 주어 호출

array\_printf("B", B, N\_DATA); // 배열 출력하는 함수에 두번째 배열이름, 배열, 배열크기 인수로 주어 호출

array\_add(A, B, C, N\_DATA); // 첫번째 배열과 두번째 배열을 더하여 월급 총액 구하는 함수에 인수를 주어 호출

array\_printf("C", C, N\_DATA); // 배열 출력하는 함수에 세번째 배열이름, 배열, 배열크기 인수로 주어 호출

return 0; // 0을 반환하여 메인 함수 끝내기

}

void array\_add(int a[], int b[], int c[], int size) // 첫번째 배열과 두번째 배열을 더해 세번째 배열에 저장하는 함수

{

// 반복문 이용하여 배열 더하기

for (int i = 0; i < size; i++) // 0부터 size만큼 반복

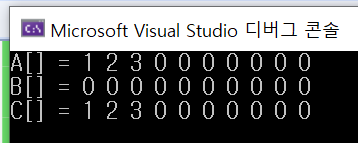
{

c[i] = a[i] + b[i]; // 0부터 차례대로 더해 세번째 배열에 저장

/\* a[0]과 b[0]을 더해 c[0]에 저장하는 것으로 시작해 size-1까지 이 과정을 반복한다.\*/

}

}



* **세율을 반영한 실수령액 구하기**

#include <stdio.h>

#define N\_DATA 10

#define RATE 0.2 // 세율 기호상수로 선언

void array\_add(double \*a, double \*b, double \*c, int size);

// 반환값이 없으며 매개변수로 포인터 3개와 정수형(배열의 크기) 변수를 받는 함수의 원형 정의

void array\_multi(double\* c, double\* d, int size);

// 반환값이 없으며 매개변수로 포인터 2개와 정수형 변수를 받는 함수의 원형 정의

void array\_printf(double\* a, double size)

// 반환값이 없으며 매개변수로 포인터, 정수형 변수를 받는 함수의 선언부

{

// 배열 출력하는 반복문

for (int i = 0; i < size; i++) // 0부터 size만큼 반복

{

printf("%3.lf ", a[i]); // 반복문을 통해 배열에 인덱스를 주어 출력

/\* 배열은 포인터와 유사하기 때문에 포인터 매개변수를 배열과 같은 형태로 사용할 수 있다.\*/

}

printf("\n"); // 줄바꿈

}

int main(void) // 반환값이 정수형이며 매개변수가 없는 메인 함수의 선언부

{

// 일차원 배열 선언 및 초기화

double A[N\_DATA] = { 180, 200, 190, 300, 150, 210, 180, 160, 350, 400 }; // 기본급 배열 선언 및 초기화

double B[N\_DATA] = { 100, 80, 50, 150, 100, 100, 90, 100, 70, 100 }; // 보너스 배열 선언 및 초기화

double C[N\_DATA] = { 0 }; // 월급의 총액 배열 선언 및 초기화

double D[N\_DATA] = { 0 }; // 세율 반영한 실제 월급 총액 배열 선언 및 초기화

printf("=======직원별 월급(단위 : 만원, 세율 : %.1lf)=======\n", RATE); // 프로그램 안내

printf(" 1 2 3 4 5 6 7 8 9 10\n"); // 직원 번호 출력

printf("기본급 = "); // 기본급 안내

array\_printf(A, N\_DATA); // 배열 출력하는 함수에 첫번째 배열이름, 배열, 배열크기 인수로 주어 호출

printf("보너스 = "); // 보너스 안내

array\_printf(B, N\_DATA); // 배열 출력하는 함수에 두번째 배열이름, 배열, 배열크기 인수로 주어 호출

array\_add(A, B, C, N\_DATA); // 첫번째 배열과 두번째 배열을 더하여 월급 총액 구하는 함수에 인수를 주어 호출

array\_multi(C, D, N\_DATA); // 세번째 배열과 네번째 배열을 인수로 주어 호출하여 실제 수령 월급 구하기

printf("세 전 = "); // 세율 반영 전 월급

array\_printf(C, N\_DATA); // 배열 출력하는 함수에 세번째 배열이름, 배열, 배열크기 인수로 주어 호출

printf("세 후 = "); // 세율 반영 후 월급

array\_printf(D, N\_DATA); // 세율 반영 후 월급 인수로 주어 출력 함수 호출

printf("==================================================\n"); // 출력 완료 안내

return 0; // 0을 반환하여 메인 함수 끝내기

}

void array\_add(double \*a, double \*b, double \*c, int size) // 첫번째 배열과 두번째 배열을 더해 세번째 배열에 저장하는 함수

{

// 반복문 이용하여 배열 더하기

for (int i = 0; i < size; i++) // 0부터 size만큼 반복

{

c[i] = a[i] + b[i]; // 0부터 차례대로 더해 세번째 배열에 저장

/\* a[0]과 b[0]을 더해 c[0]에 저장하는 것으로 시작해 size-1까지 이 과정을 반복한다.\*/

}

}

void array\_multi(double\* c, double\* d, int size) // 반환값이 없으며 실수형 포인터와 정수형 변수를 매개변수로 갖는 함수 선언부

{

// 반복문을 이용하여 세율 반영된 월급 구하기

for (int i = 0; i < size; i++) // 0부터 size만큼 반복

{

d[i] = c[i] \* (1 - RATE); // 월급에 세율을 제외한 비율을 곱하여 실수령액 구하기

}

}

