◇ 제목

◆ DESCRIPTION

◇ 제목

◆ DESCRIPTION

◇ 배열과 랜덤 함수

◆ DESCRIPTION

◆ DESCRIPTION

#include <stdio.h>

#include <stdlib.h>

int main(void) {

int list[10] = { 0 };

srand(time(NULL));

for (int i = 0; i < 100; i++)

{

int x = rand() % 10;

list[x]++;

}

int lots = list[0], number=0;

for (int n = 1; n < 10; n++)

{

if (lots < list[n])

{

lots = list[n];

number = n;

}

}

printf("가장 많이 생성된 수=%d\n", number);

}

정수형이고 크기가 10인 list배열을 생성하였다. 배열은 여러 개의 값을 저장할 수 있는 저장 공간이다. 즉 10개의 변수를 모아 놓은 것과 같은 의미가 된다. 첫 번째 배열 요소는 0으로 초기화하였고 나머지 요소는 정의하지 않았지만 배열 특징 상 모두 0이 저장이 된다 이후 for문안에서 정수형 변수i를 선언 후 0으로 초기화하였다. 반복문 블록에는 0~9의 랜덤 수를 생성해 정수형 변수x에 저장하고, 배열의 x인덱스의 값을 1씩 증가하는 소스가 있다. i가 100이 되기 전까지 반복되니 블록 소스는 100번을 반복하게 된다. 그렇게 되면 배열의 인덱스 0부터 9까지 골고루 값이 올라간다. 이후 가장 많이 생성된 수 구하는 것은 최대값 구하는 방법을 이용하였다. lots라고 하는 정수형 변수에 배열의 인덱스0 값을 저장하였고, lots값이랑 인덱스1 값을 비교해서 lots가 더 작으면 인덱스 1 값을 lots변수에 저장하였다. 이 것을 마지막 인덱스가 될 때까지 계속 반복시켰다. 그리고 조건이 참이면 number변수에는 배열 인덱스 값을 저장시켰다. 이러면 배열 인덱스 0부터 9 중 가장 값을 많이 가지는 인덱스가 출력될 수 있다.

◇ 최대값, 최소값 출력

◆ DESCRIPTION

배열을 생성하기 전에 기호상수를 정의하였다. 기호 상수는 말 그대로 상수를 기호로 표현한 것을 의미한다. 즉 상수 10을 size라고 부르기로 했다. 이후 크기가 size(10)인 정수형 list배열을 생성하였다. 리스트 배열의 요소들은 for문에 의해서 모두 랜덤 값이 채워진다. 이후 vaue1,value2라고 하는 정수형 변수를 선언하였다. 이 변수들은 각각 list의 첫 번째 요소 값으로 초기화되었다. 이후 value1과 인덱스1을 비교해서 value1값이 더 작으면 인덱스1 값을 value1에 저장하였다. 이것을 인덱스가 (size-1)까지 반복해서 최대값을 찾아내었다. 또 value2와 인덱스1를 비교해서 value2값이 더 크면 인덱스1 값을 value2에 저장하였다. 이것을 인덱스가 (size-1)까지 반복해서 최소값을 찾아내었다.

#include <stdio.h>

#include <stdlib.h>

#define size 10

int main(void) {

int list[size];

for (int i = 0; i < size; i++)

list[i] = rand();

int value1 = list[0], value2 = list[0];

for (int n = 1; n < size; n++)

{

if (value1 < list[n])

value1 = list[n];

if (value2 > list[n])

value2 = list[n];

}

printf("최대값은 %d\n", value1);

printf("최소값은 %d\n", value2);

}

◇ 배열의 초기화와 출력

◆ DESCRIPTION

기호상수 size를 정의하였다. size는 12를 의미한다. main함수안에는 크기가 size이고 정수형 배열 list를 선언하였다. 그리고 배열의 여러 값을 가지니, 중괄호를 쳐서 첫 번째 요소부터 마지막 요소까지 값을 초기화시켰다. 이후 정수형 변수 i를 1로 초기화하고 12까지 for문 블록 소스를 반복시켰다. 실제 출력은 list[i-1]로 하여서 0부터 11까지 인덱스 값을 출력하게 하였다. 인덱스 0이 배열의 첫 번째 요소를 나타내니 인덱스 11이 마지막 요소(12번째 요소)가 되기 때문이다.

#include <stdio.h>

#define size 12

int main(void) {

int list[size] = { 31,29,31,30,31,30,31,31,30,31,30,31 };

for (int i = 1;i < 13;i++)

printf("%d월은 %d일까지 있습니다.\n", i, list[i - 1]);

}

◇ 주사위 나오는 수 알아보기

◆ DESCRIPTION

6을 의미하는 기호상수 size를 정의하였다. 이후 크기가 size인 list배열을 선언하였고 모든 배열 요소의 값을 0으로 초기화하였다. for문에서는 정수형 변수 i를 선언하고 값을 0으로 초기화하였다. 이 반복문은 i가 9999가 될 떄까지 반복되고 1씩 증가시키므로 총 만 번을 시행시킨다. 블록 소스는 다음과 같다. 0부터 5까지 랜덤 값을 x변수에 저장시키고 list의 x번쨰 인덱스 값을 1씩 증가시켰다. 반복문이 끝나면 0부터 5까지 인덱스의 값들이 골고루 들어가 있을 것이다. 이후 다시 for문을 실행하는데 이번에는 0부터 size-1까지 변수i의 값(초기값 0)을 1씩 증가시켰다. 블록 소스에는 list[i]를 출력하도록 하였다. 즉 배열의 인덱스0부터 5까지 값들을 다 출력할 수 있었다.

#include <stdio.h>

#define size 6

int main(void) {

int list[size] = { 0 };

for (int i = 0; i < 10000; i++)

{

int x = (rand() % 6);

list[x]++;

}

printf("면 빈도\n");

for (int i = 0;i < size;i++)

printf("%d %d\n", i + 1, list[i]);

}

◇ 배열 비교하기

◆ DESCRIPTION

#include <stdio.h>

int main(void) {

int i, a[10], b[10], x;

printf("a배열에 들어가는 수 10개를 입력하세요:");

for (i = 0; i < 10; i++)

scanf\_s("%d", &a[i]);

printf("b배열에 들어가는 수 10개를 입력하세요:");

for (i = 0; i < 10; i++)

scanf\_s("%d", &b[i]);

for (i = 0; i < 10; i++) {

if (a[i] == b[i])

x = 1;

else {

x = 0;

break; }}

for (i = 0; i < 10; i++)

printf("%d ", a[i]);

printf("\n");

for (i = 0; i < 10; i++)

printf("%d ", b[i]);

if (x == 1)

printf("\n2개의 배열은 같음\n");

else

printf("\n2개의 배열은 다름\n"); }

정수형 변수 I,x와 크기가 10인 정수형 배열 a,b를 선언하였다. 이후 for문을 통해 인덱스가 0부터 9까지 값을 초기화시켰다. 블록 소스에 scanf문을 썼고 사용자의 입력을 10번 받아서 배열 요소의 주소에 저장시키는 방법을 택하였다. 이렇게 배열 a,b가 모두 초기화되었으면, 세 번째 for문 블록 소스는 두 배열을 비교하는 소스이다. for문에서 i의 값을 0으로 초기화하였고 i가 9가 될 떄까지 1씩 증가시켰다. 블록 소스 내용은 a배열의 인덱스i와 b배열의 인덱스i를 비교해서 같으면 x값을 1로 초기화시켰고 다르면 0으로 초기화한 후 반복을 빠져나오게 하였다. 즉 인덱스가 끝날때까지 반복시키는데, 두 배열의 인덱스가 같아야만 반복이 끝까지 진행되니 이 for문으로 두 배열이 같은지 다른지를 비교해 볼 수 있다. x=1이면 반복이 끝까지 진행됐다는 것이고 x=0이면 배열 값이 달라서 도중에 빠져 나왔다는 것이 되니 정답을 구할 수 있다.

◇ 벌레의 이동 알아보기

#include <stdio.h>

#define size 10

int main(void) {

char loads[size];

int i;

for (i = 0; i < size; i++)

loads[i] = '\_';

loads[5] = '\*';

int n = 5;

while (1)

{

for (int i = 0; i < size; i++)

printf("%c", loads[i]);

getchar();

int direction = rand() % 2;

if (direction == 0)

{

if (n == 0)

continue;

loads[n - 1] = loads[n];

loads[n] = '\_';

n--;

}

else

{

if (n == 9)

continue;

loads[n + 1] = loads[n];

loads[n] = '\_';

n++;

}

}

}

◆ DESCRIPTION

10을 의미하는 기호상수 size를 정의하였다. 문자열이고 크기가 size인 loads배열을 선언하였다, 그리고 문자열 변수 i를 선언하였다. for문에서는 i가 0부터 9까지 1씩 올리는 반복문이고 블럭 소스는 loads배열의 인덱스i를 '\_'로 초기화하는 것이다. 즉 모든 배열 요소를 '\_'로 초기화하였다. 이후 loads의 5번째 인덱스는 '\*'으로 다시 지정하였다. 밑에서는 정수형 변수 n을 선언하였고 값을 5로 초기화하였다. n은 '\*'을 가지는 인덱스 번호를 의미한다. 이후 무한루프를 돌리는데 블럭 소스 내용은 아래와 같다. 먼저 for문을 이용하여 loads배열 값을 출력하였다. 이후 getchar라고 문자열을 입력받는 함수를 호출하였다. 이 함수가 있음으로부터, 문자열을 입력받기 전까지는 다음 소스를 실행할 수 없다. 만약 문자열을 입력받았으면 정수형 변수 direction을 선언한다. 이 변수는 0혹은 1의 랜덤 값을 가지는 변수로써 반복문을 한 번 돌때마다 값이 바뀌게 될 것이다. 이후 조건문을 써서 direction값이 0이면 '\*'을 가지는 배열 인덱스의 문자를 왼쪽에 저장시켰다.(인덱스n의 문자를 인덱스n-1로 저장) 이후 원래 '\*'을 가졌던 배열 인덱스의 문자는 '\_'으로 저장시켰다. 그리고 왼쪽에 해당하는 인덱스의 번호를 n으로 지정함으로써(n에 1을 뺌)n이 '\*' 를 저장하는 배열 인덱스의 번호라는 것을 유지시켰다. 만약 '\*'값을 왼쪽으로 옮겨야 되는데 왼쪽 인덱스가 없을 수도 있다. 그래서 '\*'문자를 가지는 인덱스가 0일 때는 continue문을 써서 해당 반복을 무시하고 다음 반복을 취하게 하였다. 이로써 오류없이 '\*'을 왼쪽으로 이동시킬 수 있다. 오른 쪽으로 이동시키는 것은 direction이 1일 때이며 위와 방법을 정반대로 하면 풀린다.