Calc 함수를 완성하라

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include<stdio.h>  int calc(int a, char symbol, int b);  int main()  {  int a, b, result;  char symbol;  scanf("%d %c %d", &a, &symbol, &b);  result = calc(a, symbol, b);  printf("%d", result);  }  int calc(int a, char symbol, int b) {  int tmp;  if (symbol == '-') {  tmp = a - b;  return tmp;  }  } |

배열 실습 (평균)

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include<stdio.h>  int main()  {  float result = 0;  float arr[5] = { 0, };  int i = 0;  result = 0;  for (int i = 0; i < 5; i++) {  scanf("%f", &arr[i]);  result += arr[i];  }  printf("AV: %f", result / 5);  } |

실습 과제 1

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include<stdio.h>  int main()  {  int arr[3] = { 0, };  int i, num = 0;  int d1, d2;  int d = 0;  for (int i = 0; i < 3; i++) {  scanf("%d", &arr[i]);  }  d1= arr[1] - arr[0];  d2= arr[2] - arr[1];  if (abs(d1) < abs(d2))  d = abs(d1);  else if (abs(d1) > abs(d2))  d = abs(d2);  else  d = abs(d1);  if (abs(arr[1] - arr[0]) != d)  num = arr[0] + d;  if (abs(arr[2] - arr[1]) != d)  num = arr[1] + d;  else  num = arr[2] + d;  printf("%d", num);  } |

실습 과제2

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include<stdio.h>  #include<string.h>  int main()  {  int a = 0;  int i, max, ch;  char str[1000002] = { 0, };  int arr[26] = { 0 };  scanf("%s", str);  for (i = 0; str[i]; i++) {  if (str[i] >= 'a' && str[i] <= 'z') {  str[i] = str[i] - ('a' - 'A');  }  }  for (i = 0; str[i]; i++) {  arr[str[i] - 'A']++;  }  max = 0;  ch = 0;  for (i = 0; i<26; i++) {  if (max < arr[i]) {  max = arr[i];  ch = i;  }  }  int z, count = 0;  for (z = 0; z<26; z++) {  if (arr[z] == max)  count++;  }  if (count == 1)  printf("%c", ch + 'A');  else if (count > 1)  printf("?");  } |

텍스트이(가) 표시된 사진

자동 생성된 설명

**게임 “Wordle”**

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include<stdio.h>  #include <stdlib.h>  #include <time.h>  int main() {  char word[70][6] = { "about", "above", "abuse", "alert", "align", "alive", "alone", "amaze", "angel", "ankle", "array", "azure", "build", "cause", "curse", "cycle", "daily", "diary", "earth", "email", "entry", "every", "extra", "faith", "field", "fusil", "guess", "happy", "house", "human", "image", "juice", "knock", "knife", "kitty", "limit", "lunch", "maybe", "media", "movie", "naïve", "noise", "ocean", "olive", "opera", "other", "piano", "pizza", "quiet", "radio", "right", "sauce", "sharp", "slice", "solid", "space", "stone", "their", "today", "under", "unity", "union", "value", "video", "voice", "which", "wrong", "young", "youth", "zebra" };  char answer[6];  char input[6];  int count;  srand(time(NULL));  int i = rand() % 70;  for (int j = 0; j < 6; j++) {  answer[j] = word[i][j];  }    int try = 0;  int correct = 0;  for (try = 1; try < 7; try++) {  printf("5글자 단어를 입력하시오: ");  correct = 0;  scanf("%s", input);  for (count = 0; count < 5; count++) {  if (input[count] == answer[count]) {  printf("%c = green\n", input[count]);  correct++;  continue;  }  else if (input[count] != answer[count]) {  for (int z = 0; z < 5; z++) {  if (input[count] == answer[z]) {  printf("%c = yellow\n", input[count]);  break;  }  if (input[count] != answer[z] && z == 4) {  printf("%c = grey\n", input[count]);  }  }  }  }  if (correct == 5) {  printf("축하합니다. %d회만에 맞추셨습니다.", try);  break;  }  }  if (correct!=5)  printf("실패하셨습니다. 답은 %s입니다.", answer);  } |

**아리**

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include<stdio.h>  #include<stdlib.h>  #include<time.h>  int num = 0;  int main() {  int x = 0;  int y = 0;  int try = 0;  printf("x축의 크기 (2-9)? ");  scanf(" %d", &x);  printf("y축의 크기 (2-9)? ");  scanf(" %d", &y);    int arr[50][50] = { 0 };  srand(time(NULL));  for (int i = 0; i < y; i++) {  for (int j = 0; j < x; j++) {  int ran = rand() % 2;  arr[i][j] = ran;  }  }  for (int a = 0; a < y; a++) {  for (int b = 0; b < x; b++) {  printf("%d", arr[a][b]);  printf(" ");  }  printf("\n");  }  printf("\n");  while (1) {  printf("xy좌표(두자리 수)는? (종료를 원하면 0을 넣으세요.)");  scanf(" %d", &num);  if (num == 0) {  return 0;  }  int a = (num / 10) - 1;  int b = y - (num % 10);  int fir = b;  int sec = a;  if (fir >= 0 && sec >= 0) {  arr[fir][sec] = (arr[fir][sec] + 1) % 2;  }  if ((fir+1)<y) {  arr[fir + 1][sec] = (arr[fir + 1][sec] + 1) % 2;  }  if ((fir-1)>=0) {  arr[fir -1][sec] = (arr[fir -1][sec] + 1) % 2;  }  if ((sec+1)>=0) {  arr[fir][sec+1] = (arr[fir][sec+1] + 1) % 2;  }  if ((sec-1)<x) {  arr[fir][sec -1] = (arr[fir][sec- 1] + 1) % 2;  }  for (int a = 0; a < y; a++) {  for (int b = 0; b < x; b++) {  printf("%d", arr[a][b]);  printf(" ");  }  printf("\n");  }  int q = 0;  for (int a = 0; a < y; a++) {  for (int b = 0; b < x; b++) {  if (arr[a][b] == 1) {  q = 1;  }  }  }  if (q==0) {  printf("축하합니다!");  return 0;  }  }  } |

텍스트이(가) 표시된 사진

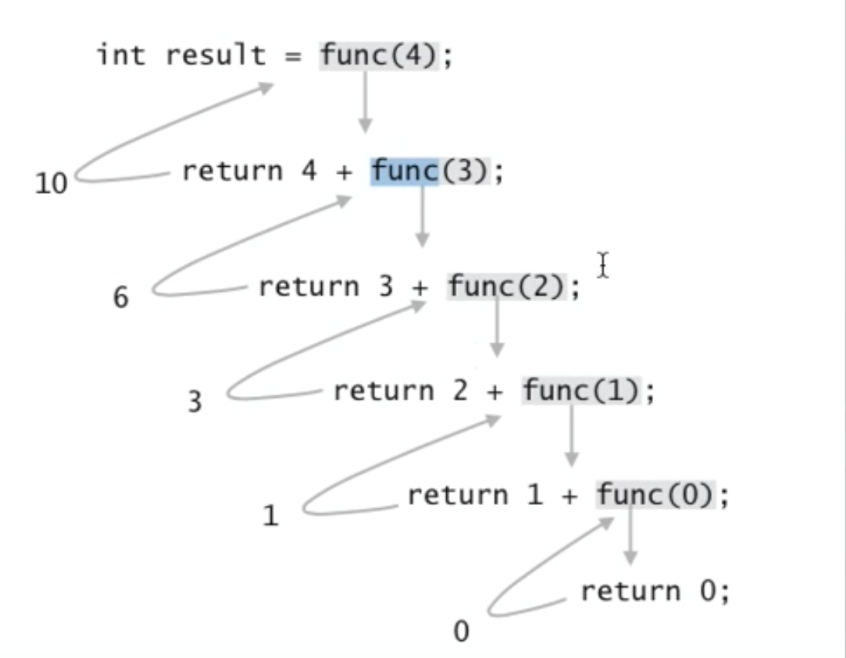
자동 생성된 설명

**재귀 함수 조사**

**재귀함수란?** – 함수가 직접 또는 간접적으로 자신을 호출하는 프로세스

함수가 반환이 되어야 끝남 / 최종적으로 마지막에 리턴이 되어야 종료, 종료 조건을 명시해야함

메모리에 복사본을 만들기 때문에 가능



주의)

함수의 정보(주소,인자)는 스택에 쌓임

재귀함수처럼 함수가 리턴되기 전에 호출하는 패턴이 무한반복

→ 스택영역이 꽉 차게 되어 스택오버플로(스택 포인터가 스택의 경계를 넘어섬) 발생

1. 난수 관련 함수에 대해 조사 및 실습(+헤더)
2. **Rand 함수**

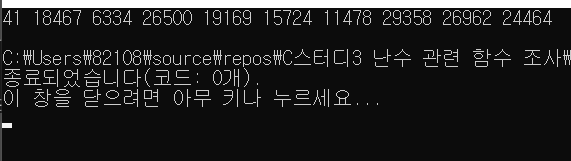
0부터 RAND\_MAX까지 범위까지 난수를 생성 (0~32767)

Rand 함수는 int형으로 랜덤한 숫자를 반환

**stdlib.h** 에 선언되어 있음.

<10개의 랜덤한 숫자를 반환>

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  #include<time.h>  int main() {  int i;  for (i = 1; i <= 10; i++)  printf("%d ", rand());  printf("\n");  } |



※ 계속 실행시켜도 매번 값이 똑같이 나옴.

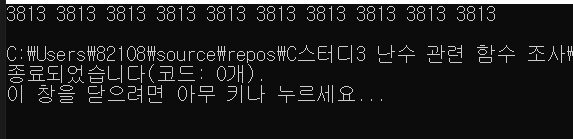
1. **Srand() 함수**

Rand 함수에 사용될 수를 초기화함

→ 난수에 발생되는 seed값을 지정함. 기본값은 1로 되어있음.

**stdlib.h** 에 선언되어 있음.

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  #include<time.h>  int main() {  int i;  for (i = 1; i <= 10; i++) {  srand(1156);  printf("%d ", rand());    }  printf("\n");  } |



※ seed에 대한 rand값이 정해져있기 때문에 일정한 rand 값이 나옴.

1. **Time 함수**

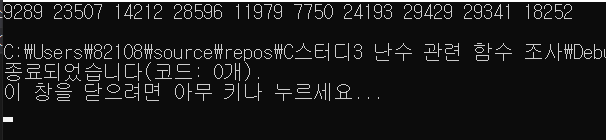
UCT 기준 1970년 1월 1일 0시 0분 0초 부터 경과된 시간을 초(sec)로 반환하는 함수 (시간에 대한 정보를 얻어오는 함수.)

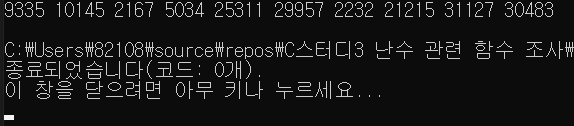
시간은 매번 바뀌니깐 srand의 seed값을 time함수를 써서 매번 바뀌게함

→ 이번 조사 과제에서는 시간이 아닌 변화하는 숫자로 사용

**time.h** 에 선언되어 있음.

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  #include<time.h>  int main() {  int i;  srand(time(NULL));  for (i = 1; i <= 10; i++)  printf("%d ", rand());  printf("\n");  } |





**+ 헤더란?**

헤더파일 – 여러 소스 파일을 컴파일하여 만든 목적파일들을 묶어놓은 파일

소스코드의 형태로 저장되어 있으며, 함수의 형태를 알기 위해 함수의 원형이 표시

#include: 컴파일하기 전에 <> 안의 것을 추가하라

1. Call by reference & call by value 조사 및 구현
2. **Call by reference (참조에 의한 호출)**

함수에서 값을 전달하는 대신 주소값을 전달하는 방식

메모리에 대한 절감

**2) Call by value (값에 의한 호출)**

함수에서 값을 복사해서 전달하는 방식

인자로 전달되는 변수를 함수의 매개변수에 복사함.

원본 값을 바꿀 필요가 없는 경우에 사용함.

|  |
| --- |
| #include <stdio.h>  //call by value  void swap1(int a, int b) {  int temp;  temp = a;  a = b;  b = temp;  }  //call by reference  void swap2(int\* a, int\* b) {  int temp;  temp = \*a;  \*a = \*b;  \*b = temp;  }  int main() {  int a = 10, b = 50;  printf("call by value\n");  swap1(a, b);  printf("결과 a : %d, b : %d\n\n", a, b);  printf("call by refernce\n");  swap2(&a, &b);  printf("결과 a : %d, b : %d\n\n", a, b);  return 0;  } |

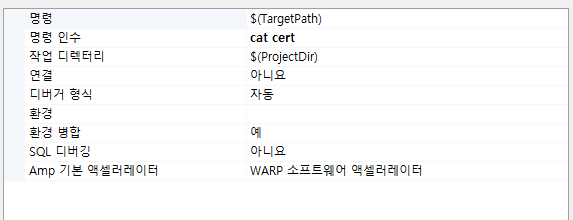
3) main 함수의 argc, argv에 대해 조사 및 실습

main 함수의 매개변수: argc, argv

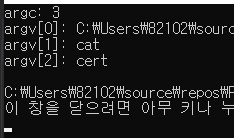
int argc: 메인함수에 전달되는 정보의 개수를 의미 / 인수 갯수를 표시함

char\* argv : 메인함수에 전달되는 실질적인 정보로, 문자열의 배열을 의미 / 인덱스가 0인 문자열은 프로그램의 실행경로로 항상 고정되어 있음

|  |
| --- |
| #include <stdio.h>  int main(int argc, char\* argv[])  {  printf("argc: %d\n", argc);  for (int i = 0; i < argc; i++)  {  printf("argv[%d]: %s \n", i, argv[i]);  }  return 0;  } |



visual studio는 명령 인수에 직접 원하는 단어를 입력해야함.



argc: 실행파일, cat, cert (3개)

4. 메모리 영역(code, data, bss, stack, heap)에 대해서 자세히 조사

c언어로 생성된 프로그램은 실행될 때 운영체제가 RAM에 메모리를 할당

→ 크게 4가지로 구성되어있음.

**1) code 영역**

실제 작동하는 명령어인  오브젝트 코드와   전역 상수가 위치

프로그램이 시작될 때 메모리에 할당되며 프로그램이 종료될 때 해제

기계어로 제어되는 메모리 영역 (Hex파일이나 BIN 파일 메모리)

**2) data 영역**

전역변수(global), 정적변수(static), 배열(array), 구조체(structure) 등이 저장

프로그램이 실행 될 때 생성되고 프로그램이 종료 되면 시스템에 반환

초기화 된 데이터는 data 영역에 저장 → 초기화 되지 않은 데이터는 BSS 영역에 저장

**3) BSS 영역**

전역 변수 중  초기화 되지 않은 변수들이 저장되는 공간

전역변수 혹은 모든 비트패턴이 0으로 초기화된 변수 (NULL로 초기화된 변수 등)

→ C언어에서는 초기화없이 정적으로 할당된 변수는 0(수치데이터의 경우)이나 NULL(포인터의 경우)로 초기화 / 0과 NULL 모두 일반적으로 컴파일러에서는 모든 비트가 0인 비트패턴으로 표현

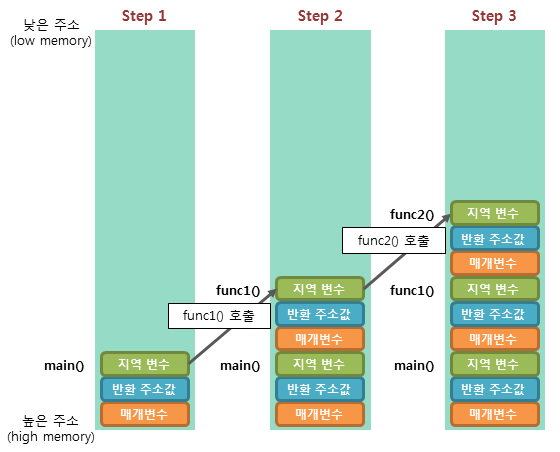
**4) stack 영역**

프로그램이 자동으로 사용하는 임시 메모리 영역

지역(local) 변수, 매개변수(parameter), 리턴 값 등 잠시 사용되었다가 사라지는 데이터를 저장하는 영역

함수 호출 시 생성되고, 함수가 끝나면 시스템에 반환

쓰레드 1개가 1개의 스택을 보유하며, 함수가 호출될 때마다 스택의 공간이 일정부분 할당



쓰레드(thread) : 어떠한 프로그램 내에서, 특히 프로세스 내에서 실행되는 흐름의 단위

**5) Heap 영역**

프로그래머가 자유롭게 할당하고 해제하는 메모리 공간

→ 동적으로 메모리를 할당함 / 런타임에 크기가 결정됨

메모리를 아래에서 위로(낮은 주소에서 높은주소)로 할당

→ 메모리를 할당한 뒤 해제를 해주지 않으면 메모리 누수가 발생

5) static변수에 대해 조사 및 실습

**static 변수:** 선언된 함수내에서만 사용이 가능(지역변수의 특성) / 한번만 초기화를 하며 프로그램이 종료될 때까지 메모리 공간에 존재 (전역변수의 특성)

tatic으로 선언한 변수는 0으로 초기화

외부참조 X (=정보은식)

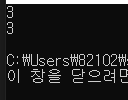
매개변수로 사용 X

**지역변수와 다른점:** 해당 함수가 종료되어도 소멸 X /

일반 지역변수는 스택(Stack)영역 에 저장 →함수 호출시에 메모리에 할당되며 함수 종료시에 메모리에서 사라짐

static 변수는 데이터(Data)영역에 저장 (+ 전역변수도 데이터 영역에 저장)

|  |
| --- |
| #include <stdio.h>  int test(void)  {  int count = 2;  count++;  return count;  }  int main(void)  {  printf("%d\n", test());  printf("%d\n", test());  return 0;  } |



6) Calling Convention: cdecl와 stdcall에 대해 조사

**cdecl:** Caller(호출자)에서 스택을 정리하는 특징, 기본 호출 규약

Win32 API에서 사용 되며, Callee(피호출자) 에서 스택을 정리

스택이 호출자에 의해 지워지기 때문에, 가변 인자를 가지는 함수를 정의할 수 있음

\_\_stdcall보다 더 큰 실행 파일을 만듦

→ 각 함수 호출마다 스택을 정리하는 코드가 추가되어야 하기 때문임.

\_\_cdecl  지정자는 변수 또는 함수 이름의 앞에 지정해야함

**stdcall:** Callee(피호출자) 에서 스택을 정리하는 것이 특징, Win32 API 함수를 호출하는 데에 사용되는 함수 호출 규약

피호출자가 스택을 정리

이 함수 호출 규약을 사용하는 함수들은 함수 원형(prototype)을 필요

비정적 클래스 함수, 함수의 정의가 클래스 블록 외부에서 정의된 경우 →

선언시에만 호출 규약 지정자를 지정하고 함수를 정의하는 곳에서 별도로 함수 호출 규약을 지정할 필요 X



7) 버블 정렬에 대해 조사

**버블 정렬:** 배열 내의 처음부터 인접한 두 데이터를 비교하며 값이 큰 데이터를 뒤로 바꾸면서 배열의 끝까지 반복하여 정렬하는 방법

EX. 오름차순으로 숫자 배열

|  |
| --- |
| #include <stdio.h>  int main() {  int arr[5] = { 7,5,6,2,9 };  int temp; //swap을 위해 선언  for (int i = 0; i < 5; i++) {  for (int j = 0; j < 4 - i; j++) {  if (arr[j] > arr[j + 1]) {  temp = arr[j];  arr[j] = arr[j + 1];  arr[j + 1] = temp;  }  }  }  for (int i = 0; i < 5; i++)  printf("%d ", arr[i]);  return 0;  } |

for문으로 arr이라는 배열의 숫자를 하나씩 비교하고 만약 이전값이 다음숫자보다 크다면 그 값을 temp라는 변수에 저장해둠. arr[j+1]을 통해 큰값이 뒤에 저장되도록 함.