Prob5

|  |
| --- |
| #include<stdio.h>  #include<string.h>  void func(char str[]) {  char tmp;  int len = strlen(str);  for (int a = 0; a < len; a++) {  for (int b = a; b < len; b++) {  if (str[a] > str[b]) {  tmp = str[a];  str[a] = str[b];  str[b] = tmp;  }  }  }  }  int main() {  char str[20];  printf("input ->");  scanf("%s", &str);  func(str);  puts(str);  return 0;  } |

텍스트이(가) 표시된 사진

자동 생성된 설명

Prob7

|  |
| --- |
| #include<stdio.h>  int snailMatrix(int b, int a, int num, int d, int e) {  int snail;  if (num == 1) {  snail = e;  return snail;  }  else {  if (d != a) {  if ((d + num - 1) != b) {  if ((d + num - 1) != a) {// +68  if (b != d) { //  snail = snailMatrix(b, a, num - 2, d + 1, e + 4 \* num - 4);  return snail;  }  }  snail = d + (num + (num \* 1));  snail = e + (snail + (snail \* 1));  return snail - b - a - 4;  }  }  snail = e + b;  return snail - (d + (d \* 1)) + a;  }  }  int main() {  int a, b, num;  printf("N : ");  scanf("%d", &num);  for (a = 0; a < num; a++) { //<+119>  for (b = 0; b < num; b++) { //<+45, 94>  printf("%4d", snailMatrix(b, a, num, 0, 1));  }  putchar(10);  }  getchar();  getchar();    } |

텍스트, 전자기기, 검은색, 키보드이(가) 표시된 사진

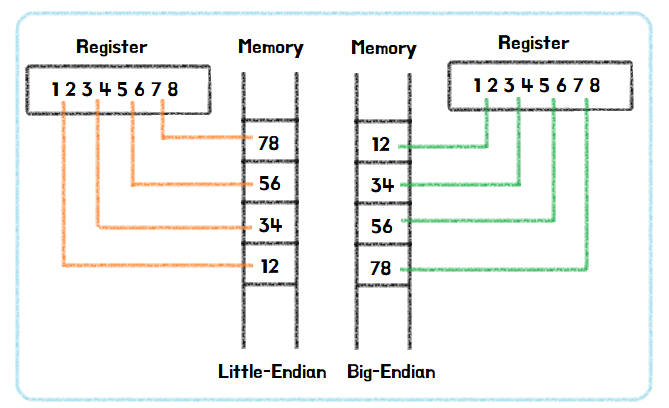
자동 생성된 설명

#리틀엔디안 빅엔디안 조사

엔디안이란?

컴퓨터의 메모리와 같은 1차원 공간에 여러 개의 연속된 대상을 배열하는 방법

바이트 순서



|  |  |  |
| --- | --- | --- |
|  | **리틀엔디안** | **빅엔디안** |
| **단위** | 작은 단위가 앞에 나옴 | 큰 단위가 앞에 나옴 |
| **0x1234** | 34 12 | 12 34 |
| **장점** | 메모리의 값을 보는데 편안함 | 별도의 계산 필요 x  (메모리에 저장된 값의 하위 바이트만 사용) |
| **숫자 비교** |  | 리틀 엔디안보다 빠름 |
| **수치 계산** | 빅 엔디안보다 빠름 |  |
| **스택** | 뒤에서부터 스택 PUSH | 앞에서부터 스택 PUSH |

Why?

**빅 엔디안** – 사람이 숫자를 읽고 쓰는 방법과 같기에 디버깅 과정에서 메모리의 값을 보는데 편안함

**리틀 엔디안** - 32비트 숫자를 리틀 엔디언으로 표현했을 때 바이트를 떼어내는 방식으로 하위 16비트, 8비트를 바로 얻을 수 있기에 별도의 계산 필요 X

숫자 비교시 가장 큰 값이 들어가는 왼쪽부터 함 / 빅엔디안은 수치를 앞에서부터 스택에 집어넣기에 리틀 엔디안보다 속도가 빠름

수치 계산시 리틀 엔디안은 가장 낮은 자리수에 있는 숫자를 계산 → 자리 올림수가 있는지 없는지 판단하기에 빅엔디안보다 속도가 빠름