제17회 교원컴퓨터프로그래밍 본선대회 문항풀이

2020.9.26.(토) 경상남도교육청

※ 문제는 다양한 방법으로 풀 수 있으며, 다음의 풀이는 여러 해법 중 하나이다.

[문제1] 자가진단 결과 프로그램

학생들의 체온 측정값을 입력받는 실수형(double) 변수 temp(단, 1.0≤temp≤100.0)을 만들고 입력받는다. 체온 측정값 temp를 등교 기준과 비교하여 체온과 등교 가능 여부를 출력한다.

- temp < 1.0 이거나, temp > 100.0 이면 체온 측정값, data error 출력
- 37.5 ≤ temp ≤ 100.0 이면 체온 측정값, stop 출력
- 36.0 ≤ temp < 37.5 이면 체온 측정값, possible 출력
- 1.0 ≤ temp < 36.0 이면 체온 측정값, retry 출력

▣ 해결방법 - 1

```
#include <stdio.h>
int main()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    double temp;
    scanf("%lf", &temp);  // 체온 측정값을 temp에 저장

    if(temp<1.0 || temp>100.0)
        printf("%.1lf data error", temp);  // 체온 측정값과 data error 출력
    else if(temp>=37.5)
        printf("%.1lf stop", temp);          // 체온 측정값과 stop 출력
    else if(temp>=36.0 && temp<37.5)
        printf("%.1lf possible", temp);     // 체온 측정값과 possible 출력
    else
        printf("%.1lf retry", temp);     // 체온 측정값과 retry 출력
 fclose(stdin);
 fclose(stdout);
 return 0;
}
```

▣ 해결방법 - 2

```
#include<stdio.h>

int main()
{
    double n;
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);

    scanf("%lf",&n);

    if(n<1.0)  printf("%.1f data error",n);
    else if( n>100.0 ) printf("%.1f data error",n);
    else if(n<36.0) printf("%.1f retry",n);
    else if(n<37.5) printf("%.1f possible", n);
    else printf("%.1f stop", n);

    return 0;
}
```

[문제2] 맞춤형 도서 추천 프로그램

자료를 2차원 배열(2*7)로 입력을 받아, 첫 번째는 대여 도서 수, 두 번째는 도서코드 값을 저장한다. 자료 입력 중에 대여 도서 수가 범위 이외의 값이면, 에러 체크를 한다. 자료 입력 완료 후에는 첫 번째로 도서 코드를 기준으로 내림차순 정렬을 하고, 두 번째로 대여 도서 수로 오름차순으로 정렬을 한다. 그리고 배열값의 7번째, 6번째, 5번째 값을 순서대로 줄바꿈을 하면서 출력한다. 마지막 행에는 입력 데이터의 오류가 있는 경우에만 data error 개수를 출력한다.

▣ 해결방법 - 1

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

        int book_recom[7][2];
        int temp[2];
        int check = 0;

        for (int i=0; i<7; i++) {
          for(int j=0; j<2; j++) {
              scanf("%d", &book_recom[i][j]);
          }

          if (book_recom[i][1] < 0)
              check = check + 1;
        }

        for (int i=0; i<7; i++) {
          for (int j=6; j>i; j--) {
              if (book_recom[j-1][0] < book_recom[j][0]) {
                  temp[0] = book_recom[j][0];
                  temp[1] = book_recom[j][1];
```

```
            book_recom[j][0]= book_recom[j-1][0];
            book_recom[j][1]= book_recom[j-1][1];

            book_recom[j-1][0]= temp[0];
            book_recom[j-1][1]= temp[1];
        }
    }
}

for (int i=0; i<7; i++) {
    for (int j=6; j>i; j--) {
        if (book_recom[j-1][1] > book_recom[j][1]) {
            temp[0] = book_recom[j][0];
            temp[1] = book_recom[j][1];

            book_recom[j][0]= book_recom[j-1][0];
            book_recom[j][1]= book_recom[j-1][1];

            book_recom[j-1][0]= temp[0];
            book_recom[j-1][1]= temp[1];
        }
    }
}

for (int i=6; i>3; i--) {
    switch(book_recom[i][0]) {
        case 1:
            printf("poem %d₩n", book_recom[i][1]);
            break;
        case 2:
            printf("novel %d₩n", book_recom[i][1]);
            break;
        case 3:
            printf("history %d₩n", book_recom[i][1]);
            break;
```

```
            case 4:
                printf("job %d₩n", book_recom[i][1]);
                break;
            case 5:
                printf("essay %d₩n", book_recom[i][1]);
                break;
            case 6:
                printf("science %d₩n", book_recom[i][1]);
                break;
            case 7:
                printf("cartoon %d₩n", book_recom[i][1]);
                break;
            default:
                printf("error %d₩n", book_recom[i][1]);
                break;
        }
       }

    if(check > 0) printf("data error %d₩n", check);


    fclose(stdin);
    fclose(stdout);

    return 0;
}
```

[문제3] 윷놀이 시뮬레이션

윷을 던진는 함수(throw) 또는 윷을 던지는 함수(throw)와 윷 형태를 카운트 하는 함수(what)을 활용한다. 함수(throw) 안에서 if 조건문 혹은 switch문을 이용하여 count 값에 따른 윷의 형태(shape)를 결정한다. 윷의 형태(shape)가 4(윷) 또는 5(모)이면, 함수(throw)를 다시 호출한다. 반환된 score값이 윷판의 길이보다 크거나 같으면 scoreA, scoreB 값을 순서대로 출력하고, 다음 행에 이긴 팀의 이름과 'win' 문구를 출력한다. 반환된 score값이 윷판의 길이보다 작으면 scoreA, scoreB 값을 순서대로 출력하고, 'continue' 문구를 출력한다. 기타 설명은 프로그램 코드의 주석문을 참고 바랍니다.

▣ 해결방법 - 1 함수, 반복문, if 조건문을 이용

```
#include <stdio.h>

int times=0;  // 윷 던지는 횟수 카운트

int throw(int score, int num, int size) // 윷 던지는 함수
{
    times++;
    int i;
    int count=0, shape, yoot[4];

    if(times>num)
        return score;

    for(i=0;i<4;i++) // 윷 형태 입력
      scanf("%d",&yoot[i]);
    for(i=0;i<4;i++) // 윷 형태 카운트
    {
        if(yoot[i]==1)
            count++;
    }
    if(count==0) // 윷 형태: '모'
        shape=5;
    else if(count==1) // 윷 형태: '도'
        shape=1;
    else if(count==2) // 윷 형태: '개'
        shape=2;
```

```
    else if(count==3) // 윷 형태: '걸'
        shape=3;
    else if(count==4) // 윷 형태: '윷'
        shape=4;
    score += shape; // 윷 형태 만큼 칸이동

    if(score>=size) // 팀이 이동한 칸이 윷판의 끝에 도달한 경우
        return score;
    else if(shape==4 || shape==5) // 윷의 형태가 '윷' 또는 '모'인 경우 한번 더 윷을 던진다.
        throw(score, num, size);
    else return score;
}

main()
{
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);

    int size, num, scoreA=0, scoreB=0; // 윷판의 길이, 윷 던지는 횟수, A팀 이동칸, B팀 이동칸 변수

    scanf("%d %d",&size,&num); // 윷판의 길이, 윷 던지는 횟수 입력

    while(times<num) // 던지는 횟수가 초과되면 윷놀이 종료
    {
        scoreA=throw(scoreA, num, size); // A팀 윷 던지기
        if(scoreA>=size) // A팀이 윷판의 끝에 도달한 경우
        {
            scoreA=size;
            printf("%d %d₩n",scoreA,scoreB);
            printf("A win");
            break;
        }
        scoreB=throw(scoreB, num, size); // B팀 윷 던지기
        if(scoreB>=size) // B팀이 윷판의 끝에 도달한 경우
```

```
        {
            scoreB=size;
            printf("%d %d₩n",scoreA,scoreB);
            printf("B win");
            break;
        }
    }
    if(scoreA<size&&scoreB<size) // 윷판에 A, B팀 둘 다 도달하지 못한 경우
    {
        printf("%d %d₩n",scoreA, scoreB);
        printf("continue");
    }
}
```

▣ 해결방법 - 2 함수, 반복문, switch문을 이용

```
#include <stdio.h>

int times=0;  // 윷 던지는 횟수 카운트

int what(int a, int b, int c, int d) // 윷 형태 함수
{
    return a+b+c+d;
}

int throw(int score, int num, int size) // 윷 던지는 함수
{
    times++;
    int i;
    int count=0, shape, yoot[4];
    if(times>num)
        return score;

    for(i=0;i<4;i++) // 윷 형태 입력
      scanf("%d",&yoot[i]);
    for(i=0;i<4;i++) // 윷 형태 카운트
```

```
    {
        if(yoot[i]==1)
            count++;
    }
    switch(what(yoot[0],yoot[1], yoot[2], yoot[3]))
    {
      case 0 : shape=5; // 윷 형태: '모'
        break;
      case 1 : shape=1; // 윷 형태: '도'
        break;
      case 2 : shape=2; // 윷 형태: '개'
        break;
      case 3 : shape=3; // 윷 형태: '걸'
        break;
      case 4 : shape=4; // 윷 형태: '윷'
        break;
    }

    score += shape; // 윷 형태 만큼 칸이동

    if(score>=size) // 팀이 이동한 칸이 윷판의 끝에 도달한 경우
        return score;
    else if(shape==4 || shape==5)
        throw(score, num, size); // 윷의 형태가 '윷' 또는 '모'인 경우 한번 더 윷을
던진다.
    else return score;
}

main()
{
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);

    int size, num, scoreA=0, scoreB=0; // 윷판의 길이, 던지는 횟수, A팀 이동칸,
B팀 이동칸 변수
```

```
    scanf("%d %d",&size,&num); // 윷판의 길이, 윷 던지는 횟수 입력

    while(times<num)
    {
        scoreA=throw(scoreA, num, size); // A팀 윷 던지기
        if(scoreA>=size) // A팀이 윷판의 끝에 도달한 경우
        {
            scoreA=size;
            printf("%d %d₩n",scoreA,scoreB);
            printf("A win");
            break;
        }
        scoreB=throw(scoreB, num, size); // B팀 윷 던지기
        if(scoreB>=size) // B팀이 윷판의 끝에 도달한 경우
        {
            scoreB=size;
            printf("%d %d₩n",scoreA,scoreB);
            printf("B win");
            break;
        }
    }
    if(scoreA<size&&scoreB<size) // 윷판에 A, B팀 둘 다 도달하지 못한 경우
    {
        printf("%d %d₩n",scoreA, scoreB);
        printf("continue");
    }
}
```

▣ 해결방법 - 3

```
#include<stdio.h>
int n, co;
int list[2];
int main()
{
    int a,b,c,d;
    int i,j,t,k;
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);
    i=0;
     scanf("%d %d₩n", &n, &co);
     for(j=0;j<co;j++){
     t=0;
     scanf("%d %d %d %d₩n", &a, &b, &c, &d);
     t=a+b+c+d;
     if (t==0) t=5;
     list[i]+=t;
     if(list[i]>=n)
         break;
     if (t<4) i=(i+1)%2;
     }
     if(list[0]>n) list[0]=n;
     if(list[1]>n) list[1]=n;
     printf("%d %d₩n", list[0],list[1]);
     if(list[0]<n && list[1]<n ) printf("continue");
     else if(list[0]>list[1]) printf("A Win");
     else if(list[0]<list[1]) printf("B Win");


    return 0;
}
```

[문제4] 가요 순위 산정 프로그램

가수들의 수를 입력받는 정수형 변수 n(단, 2≤n≤5)을 만들고 입력받는다. 가수의 이름과 가수의 디지털 음원 스트리밍 수, 음반 판매량, 방송 횟수, 시청자 투표수를 2차원 배열로 입력 받는다. 이때 가수 이름의 자료형은 문자형, 음원 성적은 정수형으로 각각 배열을 만들어 입력 받는다. 음원 성적을 구하기 위해 먼저 각 영역별 입력값의 총합을 구한다. 각 가수의 음원성적은 나눗셈과 비율을 사용하여 구하므로 자료형을 정수형에서 실수형으로 변환하여 계산한다.

점수 순으로 정렬을 할 때 가수의 이름도 함께 바꾸어야 한다. 정렬을 마치면 가수의 등수와 이름을 빈칸으로 구분하여 줄을 바꾸어 출력한다.

▣ 해결방법 - 1

```
#include <stdio.h>
int main()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    int music[5][4];
    int tot[5]={0,0,0,0,0}; //자료 초기화
    int i,j,n;
    char name[5], tempName;
    double pt[5], temp;
    scanf("%d₩n",&n);   //n에 가수의 수를 입력받음
    for(i=0;i<n;i++){
        scanf("%c",&name[i]);
        for(j=0;j<=4;j++){
                scanf("%d",&music[i][j]);
                tot[j]=tot[j]+music[i][j];  //전체 점수 합계
        }
    }
 for(i=0;i<n;i++){
    pt[i]=((double)music[i][0]/(double)tot[0]*0.65);  //정수형을 실수형으로 변환
    pt[i]=pt[i]+((double)music[i][1]/(double)tot[1]*0.05);
    pt[i]=pt[i]+((double)music[i][2]/(double)tot[2]*0.2);
    pt[i]=pt[i]+((double)music[i][3]/(double)tot[3]*0.1);
 }
 for(i=0;i<n;i++){
```

```
      for(j=0;j<n;j++){     // 점수 순으로 정렬
          if(pt[j]<=pt[j+1]){
              temp=pt[j];
              tempName=name[j];  //정렬 교환이 일어날 때 가수 이름도 함께 교환
              pt[j]=pt[j+1];
              name[j]=name[j+1];
              pt[j+1]=temp;
              name[j+1]=tempName;
          }
      }
  }
  for(i=0;i<n;i++){
       printf("%d %c₩n",i+1,name[i]); // 등수와 가수 이름 출력
  }
  fclose(stdin);
  fclose(stdout);
  return 0;
}
```

▣ 해결방법 - 2

```
#include<stdio.h>
float list[5][5];
char c[5];
float tt[5];
float re[5];
float ra[5][2];
int n;
int shh(int i)
{

    int x;
    for(x=n-1;x>i;x--)
    {
```

```
        ra[x][0]=ra[x-1][0];
        ra[x][1]=ra[x-1][1];
    }
    return 0;
}


int main()
{

    int i,j,t,k;
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);
     scanf("%d₩n", &n);
    for(t=0;t<n;t++){

     scanf("%c %f %f %f %f₩n",&c[t],
&list[t][0],&list[t][1],&list[t][2],&list[t][3]);
 //                              printf("input %c %f %f %f
%f₩n",c[t],list[t][0],list[t][1],list[t][2],list[t][3]);
          }
    for(t=0;t<n;t++){
        tt[0]+= list[t][0];
        tt[1]+= list[t][1];
        tt[2]+= list[t][2];
        tt[3]+= list[t][3];
    }
 //   printf("sum %d %d %d %d",tt[0], tt[1], tt[2], tt[3]);

    for(t=0;t<n;t++){

re[t]=((list[t][0]/tt[0])*65)+((list[t][1]/tt[1])*5)+((list[t][2]/tt[2])*20)+((list[t][3]/tt
[3])*10);

 //   printf("sum %f , %f  %f ₩n",re[t], list[t][0],tt[0]);
    }
```

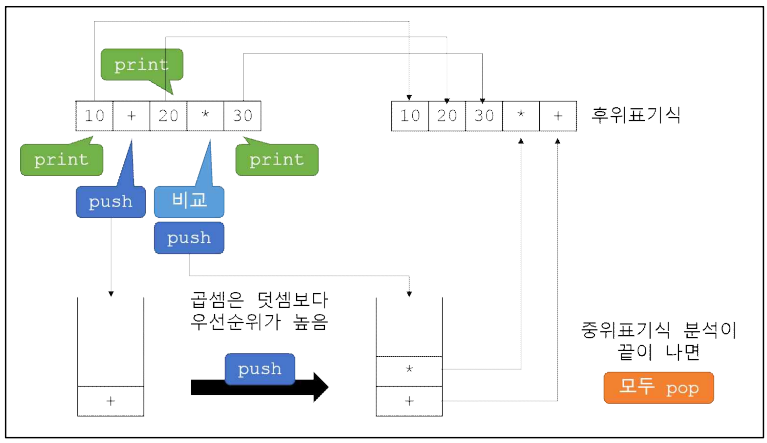
```
    for(t=0;t<n;t++){
            for(i=0;i<n;i++)
            {
                if(re[t]>ra[i][1])
                {
                    shh(i);
//                    printf("%d %d %f₩n", i, t, re[t]);
                    ra[i][0]=t;
                    ra[i][1]=re[t];
                    break;
                }



            }
               for(i=0;i<n;i++){
        k=ra[i][0];
//        printf("%d %c₩n", t+1, c[k]);
   }

    }

   for(t=0;t<n;t++){
        i=ra[t][0];
        printf("%d %c₩n", t+1, c[i]);
   }


    return 0;
}
```

[문제5] 정수형 산술·비교·논리 시뮬레이션 프로그램

입력받은 수식(중위 표기식)을 스택 자료구조를 이용하여 후위 표기식으로 변환시켜서 계산한다. 예를 들어, 10+20*30을 후위 표기식으로 변환하면 10 20 30 * +이 된다. 후위 표기식을 다시 스택을 이용하여 계산한다.

중위 표기식을 후위 표기식으로 변환하는 방법은 다음과 같다.

1. 중위 표기식을 차례대로 분석한다.
2. 피연산자는 바로 출력시킨다.
3. 처음 나타나는 연산자는 스택에 push한다.
4. 다음 나타나는 연산자부터 스택의 top의 연산자와 우선순위를 비교한다.
   1) 스택의 top의 연산자가 우선순위가 낮은 경우 push한다.
   2) 스택의 top의 연산자가 우선순위가 높은 경우
      - 스택의 top의 연산자가 우선순위가 낮을 때 까지 pop하여 출력한다.
5. 다음 연산자가 시작 괄호 '('인 경우, 스택에 push한다.
6. 다음 연산자가 끝 괄호 ')'인 경우, 스택의 top에 '('가 나타날 때 까지 계속 pop하여 출력한다.
7. 모든 분석이 완료되면, 스택에 있는 모든 연산자들을 pop하여 출력한다.
8. 연산자의 우선순위는 다음과 같다.

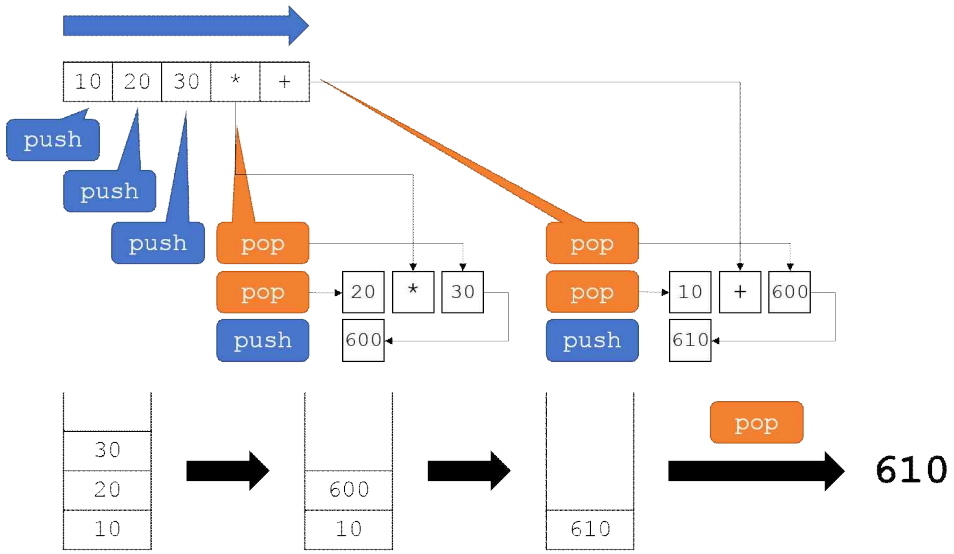


연산자	우선순위	스택 안에서의 우선순위
(	1	8
*, /, %	2	2
+, -	3	3
<, <=, >, >=	4	4
==, !=	5	5
&&	6	6
||	7	7

※ 시작 괄호 연산자는 스택 안에서의 우선순위와 스택 밖에서의 우선순위가 다르다는 점에 유의한다.

후위 표기식의 연산 방법은 다음과 같다.

1. 후위표기식을 차례대로 분석한다.
2. 피연산자는 스택에 push한다.
3. 연산자는 2개의 피연산자를 pop하여 연산한 후 결과를 push한다.
4. 후위표기식 분석이 종료되면 pop한다.



▣ 해결방법 - 1 <string.h> 사용

```
#include <stdio.h>
#include <string.h>
typedef struct {
    int n;
    char op[3];
} TOKEN;
char expr[51];
int top = -1;
TOKEN stack[100];
TOKEN post[100];
int p = 0;
void push(TOKEN *t) {
    top++;
    stack[top].n = t->n;
    strcpy(stack[top].op, t->op);
}
int pop(TOKEN *t) {
```

```
    if (top > -1) {
        t->n = stack[top].n;
        strcpy(t->op, stack[top].op);
        top--;
        return 1;
    } else {
        return 0;
    }
}
int priority(char *opt, char in_stack) {
    int p = 0;
    if (strcmp(opt, "(") == 0) {
        if (in_stack) {
            p = 8;
        } else {
            p = 1;
        }
    } else if (
        strcmp(opt, "*") == 0 ||
        strcmp(opt, "/") == 0 ||
        strcmp(opt, "%") == 0 )
    {
        p = 2;
    } else if (strcmp(opt, "+") == 0 || strcmp(opt, "-") == 0)
    {
        p = 3;
    } else if (
        strcmp(opt, "<") == 0 ||
        strcmp(opt, "<=") == 0 ||
        strcmp(opt, ">") == 0 ||
        strcmp(opt, ">=") == 0 )
    {
        p = 4;
    } else if (strcmp(opt, "==") == 0 || strcmp(opt, "!=") == 0)
    {
        p = 5;
```

```
    } else if (strcmp(opt, "&&") == 0) {
        p = 6;
    } else if (strcmp(opt, "||") == 0) {
        p = 7;
    }
    return -p;
}
void inpost(char isNum, int n, char *op) {
    if (isNum) {
        post[p].n = n;
        post[p].op[0] = 0;
    } else {
        if (op[0] == '('
            return;
        post[p].n = 0;
        strcpy(post[p].op, op);
    }
    p++;
}
int calc(char *op, int a, int b) {
    switch (op[0]) {
        case '+': return a+b;
        case '-': return a-b;
        case '*': return a*b;
        case '/': return a/b;
        case '%': return a%b;
        case '>': if (op[1]=='=') return a>=b;
                  else return a>b;
        case '<': if (op[1]=='=') return a<=b;
                  else return a<b;
        case '=': return a==b;
        case '!': return a!=b;
        case '&': return a&&b;
        case '|': return a||b;
    }
}
```

```
int main() {
    TOKEN T;
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    scanf("%s", expr);
    for (int i=0; expr[i]; i++) {
        char c = expr[i];
        char next = expr[i+1];
        if ('0'<=c && c<='9') { // 피연산자
            int n = c - '0';
            int j = i+1;
            while ('0'<=next && next<='9') {
                n = n * 10;
                n = n + (next - '0');
                next = expr[++j];
            }
            i = j - 1;
            inpost(1, n, 0);
        } else if (c == '(') { // 시작 괄호
            TOKEN t;
            t.n = 0;
            strcpy(t.op, "(");
            push(&t);
        } else if (c == ')') { // 끝 괄호
            TOKEN t;
            while (pop(&t) && t.op[0] != '(') {
                inpost(0, 0, t.op);
            }
        } else { // 연산자
            TOKEN t;
            t.n = 0;
            if ('0'<=next && next<='9' || next=='(') { // 1자리 연산자
                t.op[0] = expr[i];
                t.op[1] = 0;
                t.op[2] = 0;
```

```
        } else { // 2자리 연산자
            t.op[0] = expr[i];
            t.op[1] = expr[i+1];
            t.op[2] = 0;
            i++;
        }
        if (top == -1) {
            push(&t);
        } else if (priority(t.op, 0) > priority(stack[top].op, 1)) {
            push(&t);
        } else {
            while (priority(t.op, 0) <= priority(stack[top].op, 1)) {
                TOKEN s;
                if (pop(&s)) {
                    inpost(0, 0, s.op);
                } else {
                    break;
                }
            }
            push(&t);
        }
    }
}

while (pop(&T)) {
    inpost(0, 0, T.op);
}

for (int i=0; i<p; i++) {
    TOKEN *t = &post[i];
    if (t->op[0]) { // 연산자
        TOKEN a, b, c;
        pop(&b);
        pop(&a);
        c.op[0] = 0;
        c.n = calc(t->op, a.n, b.n);
```

```
            push(&c);
        } else { // 피연산자
            push(t);
        }
    }
    pop(&T);
    printf("%d", T.n);
    fclose(stdin);
    fclose(stdout);
}
```

▣ 해결방법 - 2 <string.h> 사용하지 않음

```
#include <stdio.h>
typedef struct {
    int n;
    char op[3];
} TOKEN;
char expr[51];
TOKEN stack[100];
int top = -1;
TOKEN post[100];
int p = 0;
void push(TOKEN *t) {
    top++;
    stack[top].n = t->n;
    stack[top].op[0] = t->op[0];
    stack[top].op[1] = t->op[1];
    stack[top].op[2] = t->op[2];
}
int pop(TOKEN *t) {
    if (top > -1) {
        t->n = stack[top].n;
        t->op[0] = stack[top].op[0];
        t->op[1] = stack[top].op[1];
```

```
        t->op[2] = stack[top].op[2];
        top--;
        return 1;
    } else {
        return 0;
    }
}
TOKEN* setToken(TOKEN *t, int n, char* op) {
    if (op == 0) { // 피연산자 토큰
        t->n = n;
        t->op[0] = 0;
    } else { // 연산자 토큰
        t->n = 0;
        t->op[0] = op[0];
        t->op[1] = op[1];
        t->op[2] = 0;
    }
    return t;
}
void inpost(TOKEN *t) {
    post[p].n = t->n;
    post[p].op[0] = t->op[0];
    post[p].op[1] = t->op[1];
    post[p].op[2] = t->op[2];
    p++;
}
int priority(char* op, char is_in_stack) {
    switch (op[0]) {
        case '(':
            if (is_in_stack)
                return -8;
            else
                return -1;
        case '*':
        case '/':
        case '%':
```

```
            return -2;
        case '+':
        case '-':
            return -3;
        case '<': // <, <=
        case '>': // >, >=
            return -4;
        case '!':
        case '=':
            return -5;
        case '&':
            return -6;
        case '|':
            return -7;
    }
}
int calc(char* op, int a, int b) {
    switch (op[0]) {
        case '+': return a+b;
        case '-': return a-b;
        case '*': return a*b;
        case '/': return a/b;
        case '%': return a%b;
        case '<':
            if (op[1]=='=')
                return a<=b;
            else
                return a<b;
        case '>':
            if (op[1]=='=')
                return a>=b;
            else
                return a>b;
        case '!': return a!=b;
        case '=': return a==b;
        case '&': return a&&b;
```

```
        case '|': return a||b;
    }
}

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "r", stdout);
    TOKEN T;
    scanf("%s", expr);
    for (int i=0; expr[i]; i++) {
        char c = expr[i];
        if ('0'<=c && c<='9') {
            TOKEN t;
            int n = c - '0';
            char next = expr[i+1];
            while ('0'<=next && next<='9') {
                i++;
                n = n * 10;
                n = n + (next - '0');
                next = expr[i+1];
            }
            inpost(setToken(&t, n, 0));
        } else if (c == ')') {
            TOKEN t;
            while (pop(&t) && t.op[0]!='(') {
                inpost(&t);
            }
        } else {
            TOKEN t;
            char next = expr[i+1];
            t.n = 0;
            t.op[0] = c;
            if ('0'<=next && next<='9' || next=='(') {
                t.op[1] = 0;
            } else {
                t.op[1] = next;
```

```
                i++;
            }
            t.op[2] = 0;
            if ( priority(t.op, 0) > priority(stack[top].op, 1) ) {
                push(&t);
            } else {
                TOKEN s;
                while (pop(&s)) {
                    inpost(&s);
                    if ( priority(t.op, 0) > priority(stack[top].op, 1) )
                        break;
                }
                push(&t);
            }
        }
    }
    while (pop(&T)) {
        inpost(&T);
    }
    for (int i=0; i<p; i++) {
        if (post[i].op[0]) {
            TOKEN a, b, c;
            pop(&b);
            pop(&a);
            push(setToken(&c, calc(post[i].op, a.n, b.n), 0));
        } else {
            push(&post[i]);
        }
    }
    pop(&T);
    printf("%d", T.n);
    fclose(stdin);
    fclose(stdout);
}
```