문항 선택

1 2 3 4 5
6 7 8 9

한 문제씩 검토
검토 완료

강의실 홈

강의정보
강의계획서

성적/출석관리
학습진도현황
오프라인출석부
성적부

수강생 알림
쪽지 보내기

기타 관리

학습활동

게시판
과제
파일
URL링크
설문조사
퀴즈

C프로그래밍 [Final-B:140]

시작 일시	2018-12-18, 18:57:52
진행 상황	종료됨
완료 일시	2018-12-18, 21:24:20
소요시간	2 시간 26 분
성적	최고 140.00점 중 **106.00**점 (**76**%)

문제 **1**

풀이 완료

총 5.00 점에서 5.00 점 할당

The lower two source codes differ only in the srand() line.

Describe/explain the role of srand().

You should also explain why time(0) is used in calling srand().

◆ Translated into Korean -->

아래 두 소스는 srand()외에 나머지 코드는 동일하다. srand() 함수의 역할에 대해 설명하라. 설명에는 srand() 함수 호출 시 쓰인 time(0)에 대한 설명도 반드시 포함되어야 한다.

Source Code #1

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

int main(void)
{
    int i,ranval;

    srand(time(0));
    for (i=0; i<10; i=i+1) {
        ranval = rand();
        printf("Random value on [0,%d]: %d\n",  RAND_MAX,
ranval);
    }

    return 0;
}
```

Source Code #2

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

int main(void)
{
    int i,ranval;

    // srand(time(0));
    for (i=0; i<10; i=i+1) {
        ranval = rand();
        printf("Random value on [0,%d]: %d\n",  RAND_MAX,
ranval);
    }

    return 0;
}
```

srand()함수는 rand()함수를 시드값을 초기화 시켜주기 위해 필요한것이다 srand()가 없으면 프로그램을 계속 돌려도 rand()함수의 값이 계속 똑같이 나올것이다
이때 srand()안에 time(0)을 넣는 이유는 시간은 계속 변하기 때문에 시드값을 변경해주기 위해서 time(0)을 넣는것이다.

댓글:

문제 **2**

풀이 완료

총 7.00 점에서 7.00 점 할당

When we execute the following program, the first value 123 of the CSV string is repeatedly extracted and the program does not terminate.

Describe the reason and how to fix it to print out all values in the CSV string.

◆ Translated into Korean -->

아래를 실행하면 CSV 문자열의 첫 값인 123만 반복해서 출력되며 프로그램이 종료하지 않는다. 이 문제의 원인을 설명하라. 그리고 CSV 문자열의 모든 수를 출력하도록 이 문제를 해결하기 위한 방법을 설명하라.

```
#include <stdio.h>
int csv_get_value(char str[])
{
    int i=0;    // index for str
    int num=0;

    if (i<0) return -1;

    while (str[i]>='0' && str[i]<='9') {
        num = num*10 + (str[i]-'0');
        i++;
    }

    if (!str[i]) i=-1;  // str[i]=='\0'
        else i++;       // str[i]==','

    return num;
}

int main(void)
{
    int num;
    char str[]="123,456,789";

    while ((num=csv_get_value(str))>=0)
                printf("%d\n",num);
    return 0;
}
```

Reason :

main 에서 csv_get_valuef를 호출 할때마다 csv_get_value 함수 안에 있는 i 변수는 automatic 이므로 함수를 호출 할때 마다 변수 i 가 계속 0으로 초기화 된다. 따라서 계속 123만 불러오는것이다

Solution :

csv_get_value 함수 안에 있는 int i=0; 을 static int i =0; 으로 바꾸게 되면 i 변수는 static 으로 바뀌게 되므로 csv_get_value 함수를 호출 할때마다 i 가 초기화 되지 않으므로 이전 i값이 유지 되어 올바르게 출력이 가능하다.

댓글:

문제 3

풀이 완료

총 8.00 점에서 4.00 점 할당

In the source code below, we can get the same execution result from "stud_printx()" and "stud_print()" while they are clearly different functions in their argument type.

Explain the differences between the 2 functions.

- Compare the amount of memory copy when they are called.
- Compare the possibility of unwanted side effect.

◆ Translated into Korean -->

아래 코드에서 stud_printx()와 stud_print()는 동일한 결과를 보이지만 분명 다른 함수이다. 두 함수가 어떻게 다른지 설명하라.

- 함수 호출 시에 필요한 메모리 복사 양을 바탕으로 두 함수를 비교하라
- 원치 않는 부가효과(side-effect) 발생 가능성 측면에서 두 함수를 비교하라

```
#include <stdio.h>
#include
typedef struct student {
    int id;
    char *pname;
    double points;
} STUD;

void stud_printx(STUD s) {
    printf("[%d:%s] = %lf₩n",
    s.id, s.pname, s.points);
}

void stud_print(STUD *ps) {
    printf("[%d:%s] = %lf₩n",
    ps->id, ps->pname, ps->points);
}

int main(void)
{
        STUD s1 = {1, "Choi", 9.9 };
        STUD s2 = {2, "Park", 0.1 };

        stud_printx(s1);
        stud_print(&s1);
        stud_printx(s2);
        stud_print(&s2);

        return 0;
}
```

c언어는 call by value 특징을 가지고 있는데 stud_printx 함수에서 구조체를 그대로 복사해 오지만 stud_print 함수는 구초체의 주소값만 가져와서 실행시키므로 stud_print함수가 메모리의 필요한 양이 stud_printx보다 더 적다.

따라서 stud_print함수의 필요한 메모리 복사 양이 적다.

side-effect 는 필요한 메모리의 양이 현재 사용할 수 있는 메모리의 양보다 더 많을때 발생하는데 stud_print 의 호출 시에 필요한 메모리 복사 양이 stud_printx보다 더 적으므로 stud_printx의 side-effect 발생 가능성이 더 높다.

댓글:

문제 4

풀이 완료

총 15.00 점에서 15.00 점 할당

Write a program that does the followings.

- Find the minimum integer n such that the sum of 1 + 2 + ... + n is no less than 1000. That is, 1 + 2 + ... + $n \geq 1000$.
- Print out "the sum" (≥ 1000) and n.

Submit your source code.

◆ Translated into Korean -->

1 + 2 + ... + n의 합이 1000을 넘는 가장 작은 합과 그 때의 n을 구하는 프로그램을 작성하라.

```
#include <stdio.h>

int segema(int k){
        int i,sum=0;
        for(i=1;i<=k;i++){
                sum =sum+ i;
        }
        return sum;
}
```

댓글:

문제 5

풀이 완료

총 15.00 점에서 15.00 점 할당

A leap year is a calendar year containing one additional day added to keep the calendar year synchronized with the astronomical year.

You can determine whether a year is a leap year or a common year using the following rules.

- If a year is not divisible by 4, it is a common year.
  Ex) 1999 -> a common year
- If a year is divisible by 4, it is a leap year except the following case.
  - If a year is divisible by both 4 and 100, but if it not divisible by 400, it is a common year. Ex) 1900 -> a common year
    - Note that If a year is divisible by 4, 100 and 400, it is a leap year. Ex) 2000 -> a leap year

Complete the isleapyear() function in the source code below which determines whether a year is a leap year (return true/1) or not (return false/0).

Submit the source code of your isleapyear();

◆ Translated into Korean -->

윤년 여부를 판단하는 isleapyear() 함수를 완성하라. 윤년이면 true/1을 아니면 false/0을 return하여야 한다.

- 4로 나눠지지 않는 해는 평년이다. 예) 1999 - 평년
- 4로 나눠지는 해는 다음 경우를 제외하고 윤년이다.
  - 4와 100으로는 나눠지지만 400으로 나눠지지 않는 해는 평년이다. Ex) 1900 - 평년
  - 참고로 4, 100, 400 모두에 의해 나눠지는 해는 윤년이다. Ex) 2000 - 윤년

Execution Result

```
1890 2015
All leap years from 1890 ~ 2015
1892 1896 1904 1908 1912 1916 1920 1924 1928 1932 1936 1940 1944 1948 1952 1956 1960 1964 1968 1972 1976 1980 1984 1988 1992
1996 2000 2004 2008 2012
```

```
#include <stdio.h>

int isleapyear(int year) {
    // Enter your code here
}

int main(void) {
        int beg, end, ny;

        scanf("%d%d",&beg, &end);

        printf("All leap years from %d ~ %d\n",
                beg, end);

        for (ny=beg; ny <= end; ny++) {
                if (isleapyear(ny))
                        printf("%d ", ny);
        }
        return 0;
}
```

```
#include <stdio.h>

int isleapyear(int year) {
        if(year%4==0){
                if(year%100==0){
                        if(year%400==0){
                if(year%100==0){
                        if(year%400==0){
                                return 1;
                        }
                        else{
                                return 0;
```

댓글:

문제 6

풀이 완료

총 20.00 점에서 20.00 점 할당

- The following program randomly generates 10 3-digit numbers and prints them out
  - All 3 digits of a number must be different. so 112, 887 are wrong values because of duplicate.
  - The first digit can't be 0, so 98 or 34 are wrong values.
- Complete the function **`generate_target_number()`** by using **`generate_a_digit()`** function.
- You are not allowed to add a new function to complete **`generate_target_number()`**.
- Submit the source code of you generate_target_number().

◆ Translated into Korean -->

10개의 3자리 난수를 생성하고 출력하는 프로그램을 작성하라. 3자리 난수는 모든 숫자가 달라야 하며 첫자리는 0이 될수 없다. 따라서 112, 887, 98, 34는 모두 잘못된 수로 포함되서는 안된다. 주어진 generate_a_digit() 함수를 이용하여 generate_target_number()를 작성하라. 새로운 함수를 추가할 수는 없다.

Execution Result
379 492 630 712 204 581 632 672 286 876

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

int generate_a_digit() {
    return (rand()%10);
}

int generate_target_number() {
    // Enter Your Code Here
}

int main(void)
{
    int nums[10];
    int i;

    srand(time(0));
    for(i = 0; i < 10; i++)
        nums[i] = generate_target_number();

    for(i = 0; i < 10; i++)
        printf("%d ", nums[i]);
    printf("\n");

    return 0;
}
```

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

int generate_a_digit() {
    return (rand()%10);
}

int generate_target_number() {
    int num,temp1,temp2;
```

댓글:

문제 7

풀이 완료

총 20.00 점에서 20.00 점 할당

strstr() is a C standard library function defined in <string.h>

char *strstr(const char* str, const char* substr);

It finds the first occurrence of the null-terminated byte string pointed to by substr in the null-terminated byte string pointed to by str. The terminating null characters are not compared.

Parameters

- str - pointer to the null-terminated byte string to examine
- substr - pointer to the null-terminated byte string to search for

Return value

- Pointer to the first character of the found substring in str, or NULL if no such substring is found. If substr points to an empty string, str is returned.

Write mystrstr() function that is equivalent to the "strstr()". In the source code below, modify mystrstr() appropriately after removing "return strstr(str, substr)". The execution result be should be the same after your modification. Note that you are not allowed to make use of "strstr()".

Submit the source code of your mystrstr().

◆ Translated into Korean -->

<string.h>에 선언된 strstr()과 동일한 기능을 수행하는 함수 mystrstr()을 작성하라. strstr()은 문자열 str에서 문자열 substr과 동일한 부분이 존재하는지를 찾고 존재할 경우 그 시작 위치/주소를 돌려주는 함수이다. 존재하지 않을 경우 null 값을 return한다. strstr()은 null문자를 비교하지는 않는다. 비교할 문자가 없을 경우 문자열 str의 시작 위치를 return한다. (아래 코드 예에서 ""로 찾는 경우에 해당). 아래 코드의 "return strstr(str, substr)"을 제거하고 동일한 기능을 수행하는 코드를 작성하라. 당연하지만 수정한 코드를 사용하여도 실행 결과는 변함이 없어야 한다.

Execution Result

```
two three
one two three
(null)
ne two three
```

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

char *mystrstr(const char *str, const char* substr) {
    // Write your own function that is equivalent to "strstr()"
    // in "string.h". You are not allowed to make use of "strstr()".
    return strstr(str, substr); // <-- Delete this line.
}

int main(void)
{
    char* str = "one two three";
    char* pos = NULL;
    char* nullstr = "(null)";

    pos = mystrstr(str, "two");
        printf("%s\n", pos ? pos : nullstr);
    pos = mystrstr(str, "");
        printf("%s\n", pos ? pos : nullstr);
    pos = mystrstr(str, "nine");
        printf("%s\n", pos ? pos : nullstr);
    pos = mystrstr(str, "n");
        printf("%s\n", pos ? pos : nullstr);

    return 0;
}
```

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

char *mystrstr(const char *str, const char* substr) {
        char *p_str,*p_substr;
        int i;
        for(p_str = str;*p_str;p_str++){
                i=0;
                for(p_substr=substr;*p_substr;p_substr++){
                        if(*(p_str+i)!=*p_substr){
```

댓글:

문제 8

풀이 완료

총 20.00 점에서 20.00 점 할당

Complete the my_atof() in the source code below that converts a string into a corresponding floating-point value(double type).

- You are not allowed to use any function in your my_atoi().
- Use "pointer based traversal" instead of "array index based traversal" in implementing my_atof(). A little penalty will be imposed if you use "array index based traversal".
- You can safely assume that the input string is terminated by '₩0' (null character).
- You can safely assume that the input string consists of only digits and no more than one ".".
  - Ex) 123.23, 3465, 27.08, 0.5

Submit the source code of your my_atof() function.

◆ Translated into Korean -->

- 문자열로 표현된 숫자를 그에 대응하는 부동소숫점 값(double type)으로 변환하는 my_atof() 함수를 작성하라.
- my_atof() 함수 안에서는 어떤 다른 함수를 호출해서도 안된다.
- 문자열을 순회할 때는 "배열 첨자"가 아닌 "포인터"를 이용하라. "배열 첨자" 기반 순회를 활용하여 구현할 경우 감점이 있다.
- 입력 문자열은 Null 문자('₩0')로 끝나며 숫자 그리고 최대 하나의 "." 로만 구성되어 있다.
  - Ex) 123.23, 3465, 27.08, 0.5

Execution Result
123.230 3456.000 0.500

```
#include <stdio.h>
double my_atof(const char *str) {
    // Enter your code
}

int main()
{
    char * digit_strs[] = {"123.23", "3456", "0.5", NULL};

    int i=0;

    while(digit_strs[i])
                printf("%.3f\n", my_atof(digit_strs[i++]));

    return 0;
}
```

```
#include <stdio.h>
double my_atof(const char *str) {
    double i_num=0;
    double f_num=0;
    int count,i;
    char *p_str;
    for(p_str=str;*p_str;p_str++){
        if(*p_str=='.'){
                break;
```

댓글:

문제 9

풀이 완료

총 30.00 점에서 0.00 점 할당

The following code sorts the array of STUD in points descending order. Complete the function find_max_stud() and sort_stud(). You are not allowed to add a new function nor to include a new header file.

Submit the source code of your find_max_stud() and sort_stud().

◆ Translated into Korean -->

다음은 주어진 STUD 구조체 배열을 points 값을 기준으로 높은 값에서 낮은 값 순으로 정렬하는 프로그램이다. 아래와 같은 정렬 결과를 얻도록 find_max_stud(), sort_stud() 두 함수를 완성하라. 단 새로운 함수를 추가 할 수 없으며 다른 헤더 파일을 추가할 수 없다.

Execution Result

```
Points Descending Order
Choi: 9.9
Moon: 9.5
Kang: 7.0
Kim: 5.0
Lee: 3.0
Jeon: 0.9
Park: 0.1
```

```
#include <stdio.h>
typedef struct student {
    int id;
    char *pname;
    double points;
} STUD;

STUD *find_max_stud(STUD *std)
{
        // Enter your code here
}

void sort_stud(STUD *std)
{
        // Enter your code here
}

int main(void)
{
    STUD pnuecs[] = { {1, "Choi", 9.9}, {2, "Park", 0.1},
                {3, "Kim", 5.0 }, {4, "Lee", 3.0 }, {5, "Moon", 9.5 },
                {6, "Kang", 7.0 }, {7, "Jeon", 0.9 }, {-1, NULL, 0 } };

    STUD *ps_array = pnuecs;
    sort_stud(pnuecs);

    printf("Points Descending Order\n");
    while (ps_array->id >= 0) {
        printf("%s: %.1f\n", ps_array->pname, ps_array->points);
        ps_array++;
    }
    return 0;
}
```

```
#include <stdio.h>
typedef struct student {
    int id;
    char *pname;
    double points;
} STUD;

STUD *find_max_stud(STUD *std)
{
}
```

댓글: