C 프로그래밍 기법 향상

김경민

함수

- 특정기능을 하는 코드 블록
  - 표준함수, 사용자 정의 함수
  - 유지보수와 가독성
  - 코드의 재사용
- 사용자 정의 함수
  - 반환자료형 함수명 (매개 변수 목록) {
    문장 …
    }

해결문제

- 자연수를 입력 받아 1에서 입력 받은 수까지의 전체합, 짝수합, 홀수합을 구하시오.
  - 한 개의 함수를 사용

해결문제

- 2보다 큰 자연수를 입력 받아서 소수인지 구분하시오.
  - 단, main 함수는 다음과 같음

```
int main()
{
        scanf("%d", &n);
        if(prime(n)) printf("prime");
        else printf("composite");
        return 0;
}
```

변수의 유효 범위(variable scope)

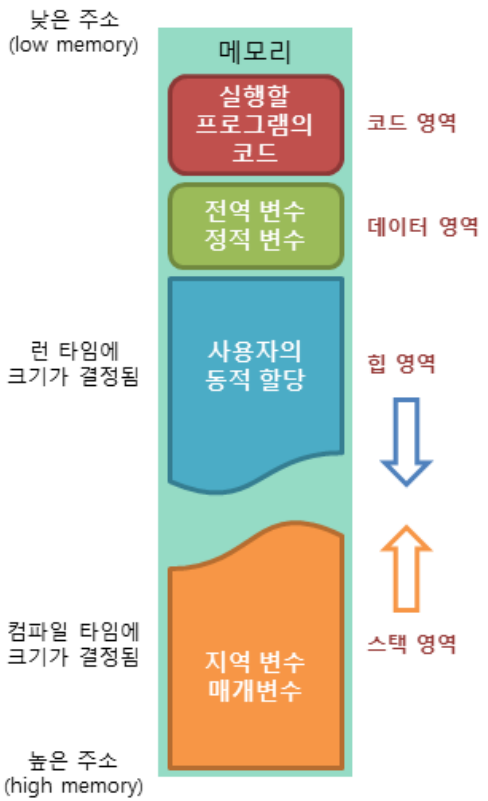
- 변수의 선언 위치에 따라 해당 변수의 유효 범위, 메모리 반환 시기, 초기화 여부, 저장되는 장소 등이 변경
- 지역 변수(local variable)
  - '블록' 내에서 선언된 변수를 의미
  - 변수가 선언된 블록 내에서만 유효하며, 블록이 종료되면 메모리에서 사라짐
  - 메모리상의 스택(stack) 영역에 저장
  - 초기화하지 않으면 의미 없는 값(쓰레기값)으로 초기화
- 전역 변수(global variable)
  - 함수의 외부에서 선언된 변수를 의미
  - 프로그램의 어디에서나 접근할 수 있으며, 프로그램이 종료되어야만 메모리에서 사라짐
  - 메모리상의 데이터(data) 영역에 저장
  - 직접 초기화하지 않아도 0으로 자동 초기화

변수의 유효 범위(variable scope)

- 정적 변수(static variable)
  - static 키워드로 선언한 변수를 의미
  - 지역 변수와 전역 변수의 특징을 모두 가지게 됨
  - 함수 내에서 선언된 정적 변수는 전역 변수처럼 단 한 번만 초기화
  - 프로그램이 종료되어야 메모리상에서 사라짐
  - 선언된 정적 변수는 지역 변수처럼 해당 함수 내에서만 접근
  - 메모리상의 데이터(data) 영역에 저장
- 레지스터 변수(register variable)
  - 지역 변수를 선언할 때 register 키워드를 붙여 선언한 변수를 의미
  - CPU의 레지스터(register) 메모리에 저장되어 빠르게 접근
  - 컴퓨터의 레지스터는 매우 작은 크기의 메모리

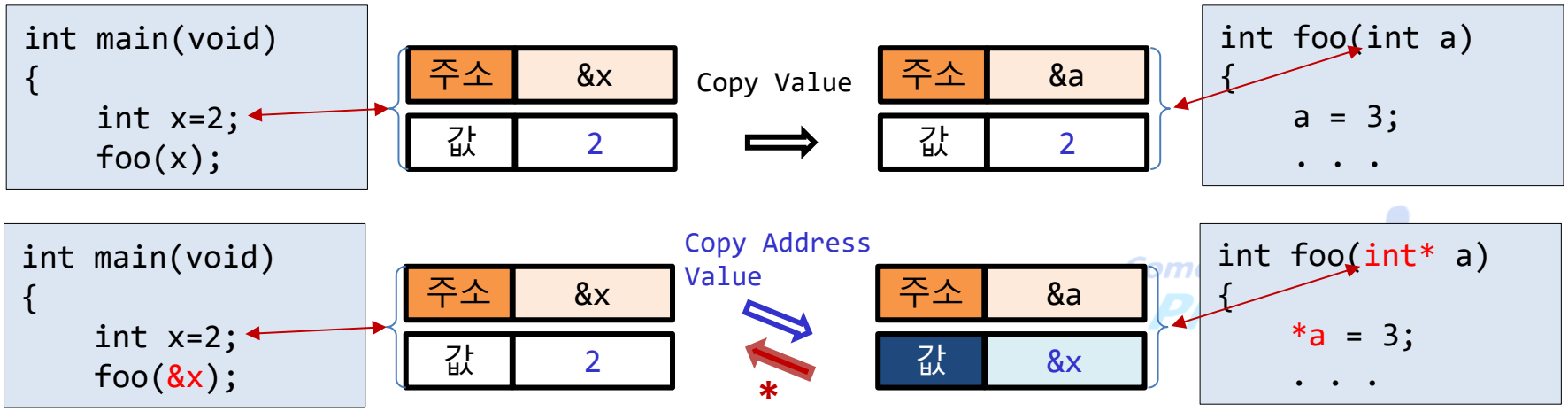
메모리의 구조

- C 프로그램이 운영체제로부터 할당받는 대표적인 메모리 공간
- 코드(code) 영역
  - 실행할 프로그램의 코드가 저장되는 영역
- 데이터(data) 영역
  - 전역 변수와 정적(static) 변수가 저장되는 영역
  - 프로그램의 시작과 함께 할당되며, 프로그램이 종료되면 소멸
- 스택(stack) 영역
  - 함수의 호출과 관계되는 지역 변수와 매개변수가 저장되는 영역
  - 함수의 호출과 함께 할당되며, 함수의 호출이 완료되면 소멸
  - Windows의 스택 크기는 1메가바이트
  - 배열일 때 대략 int numArr[1048576];
- 힙(heap) 영역
  - 사용자가 직접 관리할 수 있는 '그리고 해야만 하는' 메모리 영역
  - 사용자에 의해 메모리 공간이 동적으로 할당되고 해제
  - 메모리의 낮은 주소에서 높은 주소의 방향으로 할당



매개 변수 전달 방법

- **Call by Value**
  - 인수로 전달되는 변수가 가지고 있는 값을 함수 내의 매개변수에 복사하는 방식
  - 복사된 값으로 초기화된 매개변수는 인수로 전달된 변수와는 완전히 별개의 변수
- **Call by reference**
  - 인수로 해당 변수의 주소값을 전달
  - **인수로 전달된 변수의 값을 함수 내에서 변경 가능**



포인터

- **포인터**
  - 대상의 주소를 통해 대상을 가리키는 지시자
  - 주소를 나타내는 자료 또는 그러한 자료형
  - 포인터가 가리키는 데이터의 자료형 정보, 즉 참조 자료형 정보도 가지고 있음
- **포인터 연산자**
  - & : Address of - 참조(Referencing) 연산자, 대상의 주소를 얻어 오는 연산자
  - * : Pointer Dereference – 역참조(Dereferencing) 연산자. 주소로부터 대상 변수를 얻어오는 연산자
- **포인터 활용 – 포인터 매개 변수**
  - 포인터를 매개 변수로 전달하면 호출된 함수에서 호출한 함수의 변수 내용을 변경할 수 있음.
  - 포인터 매개 변수를 통해 함수 호출을 통해 하나 이상의 결과를 얻을 수 있음
- **배열의 이름**
  - 배열의 첫번째 요소의 주소

함수 여러 개의 리턴 값처리

- 두 수를 변경하는 swap 함수를 작성하시오.

포인터 연산과 배열

- **포인터 변수에 대한 덧셈, 뺄셈 연산**
  - 포인터 변수에 대해 정수 변수와 유사하게 + , - , ++(increment), -- (decrement) 연산을 할 수 있음
  - **정수 변수와 달리 실제 더해지거나 빼지는 값은 포인터 변수가 참조하는 자료형의 크기에 따라 달라짐**

- **배열의 이름과 주소**
  - 배열의 이름은 배열의 시작 주소를 나타내는 포인터/주소 상수
  - 상수이므로 값을 변경할 수 없음
  - 배열 이름은 포인터이므로 + 연산이 가능

- **배열 매개변수**
  - 배열을 매개변수로 사용하려면 ( ) (괄호) 안에서 매개변수 이름 뒤에 [ ] (대괄호)를 붙이거나 매개변수를 포인터로 지정

함수

- 쉼표로 분리된 문자열의 합을 구하시오.
  - 123,456,789
  - 전역변수 사용하는 경우
  - 정적변수 사용하는 경우

함수 배열 매개변수

- 1416 : 2진수 변환 프로그램을 함수로 작성

```
#include <stdio.h>
int main()
{
    int b[255] , n ;
    int i, size ;

    scanf("%d", &n) ;

    for(i=0; n > 1 ; i++) {
            b[i] = n % 2;
            n = n / 2 ;
            }

            if (n != 0) b[i] = 1;
            else b[i] = 0;
            size = i ;

            for(i=size; i>=0; i--) printf("%d", b[i]) ;


    return 0;
}
```

해결문제

- 문자열을 복사하는 함수를 작성하시오.
  - mystrcpy()는 배열형 매개 변수를 사용하며 첨자를 이용하여 문자열 복사를 수행
  - mystrcpy2()는 포인터형 매개 변수를 사용하는 함수
    - 지역 변수를 추가하지 않고 동일한 결과를 얻을 수 있도록 함수를 완성

해결문제

- 두 개의 문자열을 비교하여 모두 같으면 0을 다르면 0이 아닌 값을 return하는 mystrcmp()를 작성하시오.

문자열 함수

- #include <string.h>
  - size_t strlen(const char *str)
    - 현재 문자열(str)의 길이를 구함
    - 리턴값: 문자열의 길이
  - char *strcpy(char *dest, const char *src)
    - 문자열 복사하기
    - dest : 복사되는 변수, src : 복사할 변수
    - 리턴값: 복사된 값
  - int strcmp(const char *string1 , const char *string2)
    - string1과 string2를 비교
    - 리턴값
      - 같으면0
      - string1이 string2보다 사전적으로 앞에있으면 -1
      - 사전적으로 뒤에있으면 +1
  - char *strcat(char *dest , const char *src)
    - 문자열dest 뒤에 src를 추가

해결문제

- 문자열이 palindrome 인지를 판정하는 함수를 작성하시오.
  - int ispalindrome(char *str)
  - str : palindrome 여부를 검사할 문자열에 대한 포인터
  - return : palindrome이면 1, 그렇지 않으면 0

표준 문자 처리 함수

- <ctype.h>

함수	설명	함수	설명
int isalpha(int ch);	알파벳 여부 확인	int isalnum(int ch);	알파벳, 숫자 확인
int isupper(int ch);	대문자 여부 확인	int isprint(int ch);	출력 가능 문자 여부 확인
int islower(int ch);	소문자 여부 확인	int isgraph(int ch);	알파벳,숫자,구두점
int isdigit(int ch);	10진수 숫자 여부 확인	int iscntrl(int ch);	제어문자 여부 확인
int isxdigit(int ch);	16진수 숫자 여부 확인	int toupper(int ch);	대문자로 변환
int isspace(int ch);	공백 문자 여부 확인	int tolower(int ch);	소문자로 변환
int ispunct(int ch);	구두점 문자 여부 확인		

동적 할당(dynamic allocation)

- 프로그램이 실행되는 중에 필요한 만큼 저장공간을 힙영역에 할당하고 사용 후에 저장공간을 해제하는 것
- #include <stdlib.h>
  - void* malloc(size_t size)
    - size_t 타입은 부호없는 정수
    - 메모리 크기를 바이트 단위로 할당
    - 값들이 쓰레기값이 들어감
    - 힙 영역에 할당할 수 있는 적당한 블록이 없을 때에는 널 포인터를 반환
    - void포인터를 반환하여 개발자가 알맞은 용도로 변환하여 사용할 수 있도록 만든것
      - int *i = (int*) malloc (sizeof(int));
  - void* calloc(size_t nmemb, size_t size)
    - 메모리 크기를 바이트 단위로 할당
    - size 크기를 nmemb 개수만큼 할당
    - 0으로 초기화
  - void free(void* ptr)
    - 힙 영역에 할당받은 메모리 공간을 다시 운영체제로 반환해 주는 함수
    - 사용이 끝난 메모리를 해제하지 않아서 메모리가 부족해지는 현상을 메모리 누수(memory leak)

해결문제

- 사람의 수를 입력 받고 해당하는 사람수 만큼 키를 입력 받아서 최대값을 구하시오.

구조체

- 여러 자료형의 연속된 변수들로 구성된 집합체
  - 사용자가 C언어의 기본 타입을 가지고 새롭게 정의할 수 있는 사용자 정의 타입
    - 구조체 선언은 새로운 자료형을 정의하는 것으로 변수 선언과는 다름
  - 기본 타입만으로는 나타낼 수 없는 복잡한 데이터를 표현
    - 관련 정보를 하나의 의미로 묶을 때 사용
- 구조체 구조
  - 구조체는 하나 이상의 멤버(Member)변수로 구성
  - 각 멤버변수들은 서로 다른 자료형을 가질 수 있음

```
struct Declaration :
   struct name(optional) {
      type1 member1;
      type2 member2;
      ...
      typen membern;
   };
```

구조체 자료형 정의 및 변수 선언 방법

A

```c
#include <stdio.h>
int main(void)
{
    struct student {
        int id;
        char *pname;
        double points;
    };

    struct student s1, s2;
    ...
}
```

B

```c
#include <stdio.h>
int main(void)
{
    struct student {
        int id;
        char *pname;
        double points;
    } s1, s2;

    ...
}
```

C

```c
#include <stdio.h>
int main(void)
{
    struct {
        int id;
        char *pname;
        double points;
    } s1, s2;

    ...
}
```

typedef와 구조체 자료형

- typedef
  - 기존에 존재하는 자료형에 새로운 이름을 부여하는 것
  - 복잡한 type 이름을 간단히 표시하기 위해 주로 사용
- typedef로 구조체 자료형에 간단한 이름 부여
  - 구조체 자료형 변수/매개 변수가 자주 쓰일 경우 널리 쓰이는 기법임
  - 구조체 변수를 선언하거나 사용할 때에는 매번 struct 키워드를 사용하여 구조체임을 명시해야 함

```
struct student {
    int id;
    char *pname;
    double points;
};
typedef struct student  STUD;
```

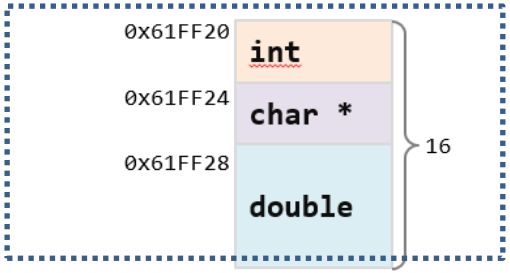
```
typedef struct student {
    int id;
    char *pname;
    double points;
} STUD;

STUD s1;
struct student s2;
```

구조체 초기화

- **구조체 변수 초기화**
  - 배열의 초기화와 유사
  - 멤버/필드 변수의 자료형과 초기화하는 값의 자료형은 일치하여야 함
- **구조체 접근**
  - 일반 변수로 선언한 구조체의 멤버에 접근할 때는 . (점)을 사용

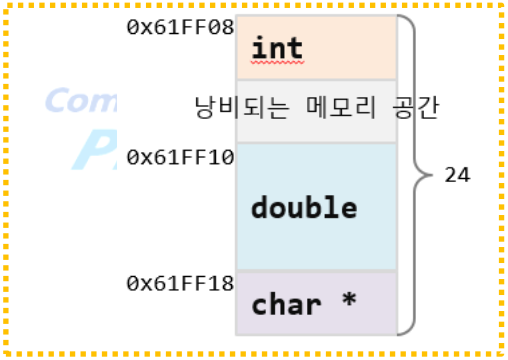
```
typedef struct student {
    int id;
    char *pname;
    double points;
} STUD;
```

```
STUD s1 = {1, "Choi", 9.9};
```




```
typedef struct student2 {
    int id;
    double points;
    char *pname;
} STUD2;
```

```
STUD2 s2 = {2, 0.1, "Park"};
```



구조체 변수에 대한 Assignment 연산

- 실제 값들이 모두 복사
- 메모리의 내용이 그대로 복사
  - 허용되기는 하지만 남용하지 않아야 하며 일반적인 경우 비추천
  - 비교) 배열 자료형에 대한 대입 연산을 지원하지 않음

구조체 포인터

- 구조체도 포인터를 선언할 수 있음
- 구조체 포인터에는 malloc 함수를 사용하여 동적 메모리를 할당 가능
- 구조체 포인터의 멤버에 접근
  - -> (화살표 연산자)를 사용

```c
#include <stdio.h>
typedef struct point {
    int x;
    int y;
} PT;

int main() {
    PT* pt1 = malloc(sizeof(PT)) ;

    pt1->x = 1;
    pt1->y = 1;

    printf("(%d, %d)\n", pt1->x, pt1->y);

    return 0;
}
```

해결문제

- point 구조체를 이용하여 두 점을 입력 받아 두 점 사이의 거리를 구하시오.

Quick Sort 함수

- Quick Sort 함수인 **qsort** 함수를 제공
  - #include <stdlib.h>

```c
#include <stdio.h>
#include <stdlib.h>    // qsort 함수가 선언된 헤더 파일

//오름 차순
int intcmp1(const void *v1, const void *v2) {
    return (*(int*)v1 - *(int*)v2);
}
//내림 차순
int intcmp2(const void *v1, const void *v2) {
    return (*(int*)v2 - *(int*)v1);
}

int main()
{
    // 정렬되지 않은 배열
    int numArr[10] = { 8, 4, 2, 5, 3, 7, 10, 1, 6, 9 };

    int i ;
    // 정렬할 배열, 요소 개수, 요소 크기, 비교 함수를 넣어줌
    qsort(numArr, sizeof(numArr) / sizeof(int), sizeof(int), intcmp1);
    for (i = 0; i < 10; i++)
    {
        printf("%d ", numArr[i]);    // 1 2 3 4 5 6 7 8 9 10
    }
    printf("\n");

    qsort(numArr, sizeof(numArr) / sizeof(int), sizeof(int), intcmp2);
    for (i = 0; i < 10; i++)
    {
        printf("%d ", numArr[i]);    // 1 2 3 4 5 6 7 8 9 10
    }
    printf("\n");

    return 0;
}
```

문자열 비교함수

```c
int compare(const void *m, const void *n) {
   return( strcmp( (char *)m, (char *)n) );
}
```

구조체의 비교함수

```c
int compareIn(const void *m, const void *n) {
    NUM *pa = (NUM *)m;
    NUM *pb = (NUM *)n;
    return pa->in - pb->in;

}

int compareOrder(const void *m, const void *n) {
    NUM *pa = (NUM *)m;
    NUM *pb = (NUM *)n;
    return pa->n - pb->n;

}
```

실습문제

- https://codeup.kr/
  - 1805 : 입체기동장치 생산공장
    - 구조체 배열
    - 구조체 동적할당
    - 정렬부분을 함수로 구현
  - 3004 : 데이터 재정렬
  - 3015 : 성적표 출력
  - 3019 : 스케줄 정리