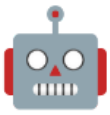

C

작성일시	@2023년 3월 4일 오전 12:41
강의 번호	더조은 컴퓨터학원
유형	강의
자료	
복습	☐

- **C 개발 환경 설정**

=> 검색창 - Visual Studio - 커뮤니티 - 데스크톱 및 모바일 C++ 사용한 데스크톱 개발 (설치 후 재부팅)

- **C 언어**

=> 예약어 = 키워드 : int, char, static, if~else, for, while, include
=> 명칭 = 식별자 : 변수명, 배열명, 함수명, 매크로명
=> 상수 = 정수상수, 실수상수, 문자상수, 문자열상수
=> 연산자

=> > : 1칸 이동할 때 마다 *2의 값 반환
=> >> : 음수의 경우 1, 양수의 경우 0으로 채움 (1칸 이동할 때 마다 2로 나눈 몫 값 반환)

※ C언어에서는 & 연산자는 주소값을 반환하는 용도로 사용한다. (Java와 Python에서는 비트연산자로 사용)

=> 설명문 = 주석 : 비실행문, 한줄주석, 여러줄주석 (//, /* 주석 */)

- **C 언어 기본 구조**

=> [함수중심 프로그래밍 (main()함수), 표준함수, 사용자정의함수)]
=> main()함수 : 프로그램 실행의 시작과 끝
=> print()함수 : "문자열상수" 또는 인수의 값 화면 출력

```
#include <studio.h>
printf("출력 양식", ---)
```

```
ex)
#include <stdio.h>                          ※stdio.h 표준입출력 헤더파일, printf()함수 사용

  void main() // C프로그래밍 "실행"의 시작
  {                              // 실행문;
    printf("여러분, 합격합시다!\n");        ※\n : new line 줄바꿈
    printf("Let's study Hard~\n")
    printf("알고리즘 : 25점\n");
    printf("알고리즘 : %d점\n",25);           ※'%d' : 10진수 , 출력결과 -> 알고리즘 : 이십오점
    printf("알고리즘 : 25점+5점\n");        ※ 출력결과 -> 알고리즘 : 이오+오점
    printf("합격점수 : %d\n",100);          ※ 출력결과 -> 정수상수 100
    printf("원주율 : %f\n",3.14);           ※ 출력결과 -> 실수상수 3.14
    printf("학점 : %c\n",'A');                ※ 출력결과 -> 문자상수 'A'
    printf("메세지 : %s\n","PASS");         ※ 출력결과 -> 문자열상수 "PASS"
    printf("%d + %d = %d\n",10,20,10+20);   ※ 출력결과 -> 10+20=30
  }                                           ※ C프로그래밍 "실행"의 종료
                                              ※ %o(8진 정수), %x(16진 정수), %u(부호 없는 10진
 정수), %e(지수)
```

- **C 프로그램 상수**

=> 10진수 123의 표기
=> 8진수 : 0123
=> 16진수 : 0x123

=> 10진수 3.14의 표기 31.4x10^1 -> 31.4e+1
=> 'A' --> 65
=> 문자열 끝에 null문자(\0) 추가

- **변수 4요소**

=> 시작 주소, 이름, 자료형, 값
=> 변수의 **선언문** : int A;
=> 변수의 **대입문** : A = 10;
=> 변수 **선언**과 동시에 **대입** : int A = 10

- **변수의 선언문(자료형)**

=> **정수형** : int(4), short, long, usinged
=> **실수형** : float(4), double(8), long double
=> **문자형** : char(1), usined char
=> **열거형** : enum
=> **형없음** : void
※ 변수명 정의 규칙 : 영문, 대소문자, 숫자를 섞어 명명, 대소문자 구별, 언더바(_)만 사용가능

- **scanf 함수** : 콘솔화면에서 키보드로부터 자료를 주어진 입력양식으로 입력시키는 함수

=> scanf("%d",&Age);
=> %d:10진 정수 변환
=> %f:부동소숫점 형식 변환
=> %c:한문자로 변환
=> %s:문자열로 변환

- **연산자**

=> 산술연산자 : + - * / % ++ --
=> 관계연산자 : > < >= <= == !=
=> 논리연산자 : && || !
=> 대입연산자 : += -= *= /= <= /= %=
=> 조건연산사 : ? :
=> 비트연산자 : & | ^ ! << >>
=> 기타연산자 : sizeof() cast & *

※ 연산처리순서 : 단항연산(1차) → 이항연산 → 삼항연산 → 산술연산 → 관계연산 → 논리연산

- **연산자 유형**

1) 이항연산자

```
#include <stdio.h>

    void main()
    {
      int x, y;
      x = 5;
      y = 2;
      printf("덧 셈 : 5+2=%d \n" , x+y);
      printf("뺄 셈 : 5-2=%d \n" , x-y);
      printf("곱 셈 : 5*2=%d \n" , x*y);
        ※ printf("나눗셈 : 5/2=%d \n" , x/y);
      printf("나머지 : 5%2=%d \n" , x%y);
    }
```

2) 단항연산자

```
#include <stdio.h>

    void main()
    {
```

```
      int x, y;
      x = 10;
      y = ++x;                          ※ x변수를 1씩 증가 시킴 (전위)
      printf("x : %d \n" , x);          ※ 결과 11
      printf("y : %d \n" , y);
      x = 10;
      y = x++                           ※ x변수를 1씩 증가 시킴 (후위)
      printf("x : %d \n" , x);          ※ 결과 11
      printf("y : %d \n" , y);          ※ 결과 10
    }
```

3) 관계연산자 --> 참과 거짓 결과 출력(논리값)

```
#include <stdio.h>

    void main()
    {
     int x, y;
     x = 5;
     y = 2;
     printf("5>2 : %d \n" , x>y);          ※ 결과 1
     printf("5<2 : %d \n" , x<y);          ※ 결과 0
     printf("5==2 : %d \n" , x==y);        ※ 결과 0
     printf("5!= : %d \n" , x!=y);         ※ 결과 1
}
```

4) 논리연산자(Byte 단위) / 처리순서 : 논리부정(NOT) -> 논리곱(AND) -> 논리합(OR)

```
#include <stdio.h>

    void main()
    {
      int month, day, birthday, jan;
      month = 1;
      day = 10;
      birthday = month==8 && day==19;             ※ 결과 0 && 0 = 0
      printf("birthday : %d \n" , birthday);      ※ 결과 0
      jan = month==1 || day==1;                   ※ 결과 0 || 1 = 1
      printf("jan : %d \n" , jan);                ※ 결과 1
    }
```

5) 대입연산자 -> 복합연산자

```
#include <stdio.h>

    void main()
    {
      int a = 10;
      a = a + 1; printf("%d\n" , a);
      a += 1; printf("%d\n" , a);
      ++a; printf("%d\n" , a);
```

```
      a++; printf("%d\n" , a);
    }
```

6) 조건연산자

- 항1 ? 항2 : 항3

=> 참일경우 항2로
=> 거짓일경우 항3으로

```
#include <stdio.h>

    void main()
    {
      int x, y;
      x = 5;
      y = 2;
      max = (x>y) ? x : y;                ※ 5 > 2 ? 5
      printf("큰값 : %d \n" , max);       ※ 결과 5
    }
```

7) 비트연산자 & 기타연산자 --> 메모리의 크기 설정

- 비트연산자 : & | ^ ~ << >>
- 기타연산자 : sizeof() cast & *

```
#include <stdio.h>

    void main()
    {
      printf("int형의 크기 : %d바이트\n" , sizeof(int));          ※ 결과 4
      printf("float형의 크기 : %d바이트\n" , sizeof(float));      ※ 결과 4
      printf("double형의 크기 : %d바이트\n" , sizeof(double));    ※ 결과 8
      printf("char형의 크기 : %d바이트\n" , sizeof(char));        ※ 결과 1
     }
```

8) 형변환연산자(cast)

```
#include <stdio.h>

    void main()
    {
      int x, y;
      x = 5;
      y = 2;
      printf("x/y : %d\n" , x/y);                              ※ 결과 5/2 -> 2
      printf("x/y : %f\n" , (double)x / (double)y);            ※ 결과 (double)5
      (double)2
    }
```

- **구조화 프로그램** : 순차제어구조, 선택제어구조, 반복제어구조

1) 순차구조

- int a; ※ **변수 선언문;**
- a = 10 + 20 ※ **변수 대입문;**

- **제어구조**

1) 순차구조
2) 선택구조 : if문 , switch~case문
3) 반복구조 : while문 , do~while문 , for문
4) 제어명령문 : break; , continue; , goto;

- **if문 : 단순if , if~else , 중첩if**

```
#include <stdio.h>

    void main()
    {
      int month, day, age;  age=20;
        printf("날짜 입력 > 월(1~12) :");
        scanf("%d",&month);
        printf("날짜 입력 > 일(1~31) :");
        scanf("%d",&day);

      if( month==1 && day==1 )
        age = age + 1
      printf("나이 : %d\n" , age);
    }


ex1)
#include <stdio.h>

    void main()
    {
     int x, y, nax; x=5; y=2;     ※ max = (x>y) ? x : y;

     if (x > y)
          max = x;
     else
          max = y;
     printf("큰값 : %d \n" , max);
    }

ex2)
#include <stdio.h>
```

```
    void main()
    {
     int number;
     printf("정수입력 :");
     scanf("%d" , &number);
     if(number>0)
         printf("positive Number\n");
     else if(number==0)
         printf("ZERO\n");
     else
         printf("Negative Number\n");
  }
```

- **switch ~ case문**

=> 행번호 x , 코드위치 : "레이블”

```
#include <stdio.h>

  void main()
  {
  int season;
  printf("계절 구분 > 봄(1),여름(2),가을(3),겨울(4) : ");
  scanf("%d",&season);
  switch(season)
  {
   case 1: printf("봄 소풍 가세요~\n"); break;
   case 2: printf("바다로 갈까요?\n");  break;
   case 3: printf("단풍구경 갑시다.\n:"); break;
   case 4: printf("스키장, 어떠세요\n"); break;
  }
```

- **while문**

```
#include <stdio.h>

  void main()
  {
     int i;
     i = 0;                          ※ 변수 i : 반복용 제어변수 (1~5, +1)
    while(i<5)
  {
       printf("합격!\n");          ※ 반복대상
      i++;                           ※ i=i+1; (증감값)
  }
  }

문제 1) 1 ~ 10까지 합계

  #include <stdio.h>

  void main()
  {
         int i; int sum = 0;
```

```
        i = 1;
        while(i<=10) // while(i<11)
        {
               sum += i;  // sum=sum+i;
               i++;          // i=i+1;
        }
        printf("1부터 10까지의 합 : %d\n" , sum);
  }
```

```
문제 2) 구구단 2단

1. while문

  #include <stdio.h>

  void main()
    {
      int i;
      i = 1; // for문 변경 : for(i=1; 1<10; i++)
        printf("==구구단 : 2단 출력==\n");
        while(i<10)     // while(i<=9)
        {
              printf("%d * %d = %2d\n" , 2 , i , 2*i);
              i++
        }
  }

2. for문

  #include <stdio.h>

  void main()
  {
        int i;
        printf("==구구단 : 2단 출력==\n");
        for(i=1; 1<10; i++) // for(초기화 ; 조건식 ; 증감값)
        {
              printf("%d * %d = %2d\n" , 2 , i , 2*i);
        }
  }
```

※ 무한반복

while(1) for(;;){

{

반복대상 반복대상

```
문제 3) while문 무한반복

  #include <stdio.h>

  void main()
```

```
  {
        int num, sum=0;
     while(1)
  {
           printf("정수 입력(끝:0)···?");
           scanf("%d",&num);
           if(num==0)
              break;
           sum += num; // sum=sum+num
        }
        printf("입력한 정수의 합계 : %d\n" , sum);

  }
```

```
※ goto문 (C언어에서만 존재)

  #include <stdio.h>

  void main()
  {
    int i;
    START:
      printf("문자 하나를 입력 (Q:종료) :");
            scnaf("%c", &ch);
        if(ch=='Q')
      goto END;
    else
                goto START;
    END:   printf("종료\n");
  }
```

- **C언어 & 순서도**

1) 반복구조

```
1 ~ 10까지의 합

      #include <stdio.h>                ※ 결과출력시에만 사용

      void main()                       ※ 순서도의 START 부분
      {
        int SUM = 0;                    ※ 변수 초기화 및 선언
        int N = 1;                      ※ 변수 N : 1~10, +1
      do
      {
        SUM = SUM + N;                  ※ SUM += N; (+= : 누적표현)
        N = N + 1;                      ※ N += 1; , N++; , ++N;
      } while (N <= 10);
        printf("%d\n", SUM);
        return 0;
    }


      #include <stdio.h>
```

```
      void main()
      {
        int SUM = 0;                    ※ 변수 초기화 및 선언
        int N = 1;                      ※ 변수 N : 1~10, +1
      while(1)                          ※ 0을 제외한 항상 참에 해당하는 조건식 , 무한반복
      {
        SUM = SUM + N;                  ※ SUM += N; (+= : 누적표현)
        N = N + 1;                      ※ N += 1; , N++; , ++N;

      if(N <= 10) break;
      }

        printf("%d\n", SUM);
        return 0;
    }
```

2) 변수의 주요기능

```
1. 홀수 덧셈 , 짝수 뺄셈

#include <stdio.h>

       int main()
       {
        int SUM = 0; int N = 1; int SW = 1;         ※ 변수 SUM : 결과(누적) , 변수 SW : 교대
       do {                                         ※ 변수 N : 1) 반복제어 , 2) 항
          SUM = SUM + (SW * N);                     ※ 선처리 // SUM = SUM + (N*-1); , 참 :
누적
        if (N < 100) {                              ※ 후검사
          N = N + 1;                                ※ N+=1; , ++N , N++
          SW = -SW;                                 ※ SW = SW * -1;
        }
        else
        {
        break;                                      ※ 거짓 : 반복종료
        }
        } while(1);                                 ※ while(항상 참) , 무한반복
          printf("(+1)+(-2)+(+3)+(-4)...+(-100)%d\n" , SUM);
          return 0;
      }

2. 임시 변수 TEMP를 이용한 변수 교환(스와핑)

      #include <stdio.h>

       main()
       {
        int A = 10;
        int B = 20;
        int TEMP;
          printf("교환전 : %d %d\n" , A , B);
        TEMP = A;
        A = B;
        B = TEMP;
          printf("교환후 : %d %d\n" , A , B);
```

```
    return 0;
  }
```

3) 기본수열과 피보나치 수열

```
1. 1부터 100까지 자연수 합

    #include <stdio.h>

    main()
    {
      int SUM = 0;
      int N = 1;
    while(1)                          ※ 유한반복 : While(N<=100)
    {
      SUM = SUM + N;                  ※ SUM += N;
    N = N + 1                         ※ N += 1; , ++N , N++

    if(N > 100) break;                ※ 유한반복 시 break x
    }
      printf("1~100까지의 합계 : %d\n" , SUM);
    return 0;
  }

2. 등차수열 : 2,8,14,20,26,32 .. 200번째 숫자까지의 합계

    #include <stdio.h>

    main()
    {
        int A = 2;                    ※ 수열의 초항
        int D = 6;                    ※ 수열의 공차(차이)
        int S = A;                    ※ 200번째 항까지의 합
        int N = 2;                    ※ 등차수열의 항 순서 , 항수 , 반복횟수
        int AN = 0;                   ※ 항
    while(1) {
          AN = A + (N-1) * D;         ※ AN = A + 6;
          S = S + AN;                 ※ S += AN;
          N = N + 1;
    if(N > 200) break;
    }
      printf("등차수열의 합 : %d\n" , S);
    return 0;
  }

3. 등비수열 : 2,6,18,54,162 .. 100번째 숫자까지의 합계

    #include <stdio.h>

    main()
    {

      int R = 3;                      ※ 수열의 공비 (*3)
      int A = 2;                      ※ 수열의 초항
      int S = A;                      ※ 100번째 항까지의 합 보관 변수
      int N = 2;                      ※ 등비수열의 항 순서
```

```
      while(1) {
            A = A * R;                  ※ A *= R;
            S = S + A;                  ※ S += A;
            N= N + 1 // N++

      if(N > 100 ) break;               ※ 100번째 항까지
      }
        printf("2+6+18-- : %d\n" , S);
      }
```

```
4. 피보나치수열 : 1,1,2,3,5,8, .. 수열에 대하여 100번째 항까지 합

      #include <stdio.h>

      int main()
      {
        int A = 1, B = 1, S =  A + B;             ※ s=2
        int N = 2;                                ※ 선처리 후증가
        int C = 0;                                ※ 세번째항 = 피보나치항

      while(1) {
            C = A + B;
            S += C;
            A = B;                                ※ 다음 항을 만들기 위한 선행 작업
            B = C;
            N++

      if(N > 100) break;
      }
        printf("피보나치수열의 합 : %d\n" , S);
      return 0;
    }
```

4) 기본 수학 (합계, 평균, 개수, 최대값, 최소값)

```
1. 100명 학생 중 영어 시험 성적이 80점 이상인 학생 수 구하기

      #include <stdio.h>

      int main()
      {

        int JUMSU[10] = {70, 60, 55, 90, 85, 75, 80, 100, 95, 45};
        int CNT = 0;
        int i = 0;                      ※ 변수 i : 1)반복제어변수 2)배열첨자
                                        ※ 1차원 배열의 첫 요소의 첨자는 0 index
      while(1) {
      if (JUMSU[i] >= 80)
          CNT++;
          i++
      if (i >= 10) break;                         // 10명까지
      }
        printf("영어점수 80점 이상인 학생수 : %d\n" , CNT);
      return 0;
    }
```

1-1. for문 활용

```
      #include <stdio.h>

      int main()
      {

        int JUMSU[10] = {70, 60, 55, 90, 85, 75, 80, 100, 95, 45};
        int CNT = 0;
        int i

      for (i=0; i<=9; i++)
      {
      if (JUMSU[i] >= 80)
      {
          CNT++;
      }
    }
        printf("영어점수 80점 이상인 학생수 : %d\n" , CNT);
      return 0;
      }
```

2. 200명 학생, 영어 시험 만점 학생들 중에서 가장 높은 수학 시험 점수 구하기(최대값)

```
      #include <stdio.h>

      int main()
      {
        int M = 0; int i = 0;           ※ M : 수학점수의 최대값
        int ENG[10] = {70, 60, 55, 90, 85, 75, 80, 100, 95, 45};
        int MATH[10] = {90, 45, 60, 77, 85, 65, 80, 95, 80, 55);

      while(1) {

      if (ENG[i] == 100)
      {                               ※ 참 , 영어접수가 100점인 경우
                }

      if (MATH[i] > M)
      {
        M = MATH[i];
      }
        i++

      if (i >= 10) break;
      }
        printf("영어100점수, 수학점수의 최고점수 : %d점/n" , M);
      return 0;
    }
```

3. 한달 30일, 일일 통화시간이 200초 이하이면 무료, 총 통화시간 제외, 평균 통화시간 구하기

```
      #include <stdio.h>

      int main()
      {

        double T[30] = {184, 240, 9, 235, 333, 295, 20, 38, 329, 32, 350, 59, 313, 24
 ...}
```

```
  int Sum = 0; int N = 0; int i = 0;
  double Avg;                       ※ flot Avg;

while(1) {

if (T[i] > 200) {
    Sum += T[i];                    ※ Sum = Sum + T[i]
    N++
}
    i++;
if (i >= 30) break;
}
  double Avg = (double) Sum / N;
  printf("평균 통화시간(초) : %f초\n" , Avg);    ※ f : 소수이하 6자리
return 0;
}
```

5) 소수와 소인수

```
1. 소수 판별

      #include <stdio.h>                ※ printf()함수를 위해 사용
      #include <math.h>                 ※ 표준수학함수 기능 : sqrt()함수
       int main()
       {

        int P = 2;                      ※ 소수의 최대값
        int N = 3;                      ※ 3~100 자연수(판별대상)
        int i = 2;                      ※ N / i <-- 제수(2~N의 제곱근)
        int R;                          ※ 나머지
        int M;

      while(1) {
            int M = int(sqrt(double(N)));
      for (int i = 2; i <= M; i++) {
      int R = N % i;                    ※ & 나머지 연산자
      if (R == 0) break;                ※ 참_고구 아님
      if (i == M) P = N;                ※ 참_최대소수 갱신
      }
        N++
      if ( n > 100 ) break;
      }
        printf("1~100까지의 가장 큰 소수 : %d\n" , P);
      return 0;
    }

2. 소인수 분해

      #include <stdio.h>
      #include <math.h>

      int main()
      {
        int A[20];                      ※ 1차원 배열 : 소인수
        int N = 20;                     ※ 소인수 분해 대상 자연수 // scanf("%d" , &N);
        int T = 0;                      ※ 배열 A의 첨자(인덱스)
        int P = 2;                      ※ 소인수
```

```
        int J;

        // 소인수 구하기
      do { if (N >= 2 {
        for (P = 2; P <= N; P++) {
        if (N % P ==0) break; }
          A[T] = P;                        ※ P : 소인수
          N = N / P;
          T++;
        }

      else return 0;                       ※ 메인프로그램 종료

      } while (N != 1);                    ※ N이 1이 아닐때까지 반복

                                           ※ 소수 또는 소인수 분해 출력
      if (T == 1) printf("소수\n");
      else {
      for (int J = 0; J < T-1; J++) {
          printf("%d * " , A[J]); }
          printf("%d\n" , A[T-1]);
      }
      return 0;
    }
```

6) 배수와 공배수 & 약수와 완전수

=> 완전수 : 자연수 중에서 자기자신을 뺀 약수들의 합이 자기자신과 같아지는 수
=> ex) 28 --> 약수 (1,2,4,7,14,28) --> 28을 뺀 나머지 수의 합이 28임

```
1. 3의 배수이면서 4의 배수인 수의 개수 : 배수 판별

      #include <math.h>

      int main()
      {
        int A[10] = {21, 17, 4, 51, 24, 75, 40, 27, 48, 72};
        int CNT = 0;
        int i = 0;
        int N3;
        int N4;
        int N;                                ※ 공배수
      do {int N3 = A[i] % 3;                  ※ 3의 배수
        int N4 = A[i] % 4;                    ※ 4의 배수
        int N = N3 + N4;

      if (N == 0)
        CNT++; i++;

      } while (i < 10);
          printf("3의 배수이면서, 4의 배수의 갯수 : %d개\n" , CNT);
          return 0;
}
```

```
      ** do { int N3 = A[i] % 3;
          int N4 = A[i] % 4;
          int N = N3 + N4;

2. if(!(A[i]) % 3 && !(A[i] % 4))
   * 완전수 : 4부터 500까지의 자연수 중에서 완전수를 찾아 출력하고 그 개수 출력

      #include <math.h>

      int main()
      {
        int TN = 0;                              ※ 4~500까지의 완전수 개수
        int N;
        int SUM;
        int J, K, R;

      for (int N = 4; N <= 500; N++) {
        SUM = 0;
        K = N / 2;                               ※ 정수 / 정수 : 정수
                                                 ※ N의 약수(J)를 구하고, 약수들의 합(SUM) 누적

      for (int J = 1; J <= K; J++) {
      if (N % J == 0)

        ※ N의 약수(J)
        SUM += J;
      }

        ※ N의 완전수 판별
      if (N == SUM) {

        ※ N은 완전수
        printf("완전수 : %d\n" , N);
          TN++;
      }
    }
        printf("%d\n" , TN);
  }
```

7) 유클리드호제법 & 근사값

```
* 유클리드호제법 : 최대공약수(GCD)와 최소공배수(LCM)

      #include <math.h>

      int main()
      {
        int X = 60, Y = 124;
        int A, B;
        int TEMP;                          ※ 교환 임시변수

        printf("두 정수 입력 : ");          ※ 두 정수 입력
        scanf(%d %d , &A, &B);
        X = A; Y = B;

      if (X < Y) {                         ※ X : 큰수 , Y : 작은수
        int TEMP = X;
```

```
        X = Y;                              ※ 교환(SWAP)
        Y = TEMP;
      }

      ※ 나머지 구하기
      for (;;) {                            ※ while(1) , for문에서 무한반복 코드
        int M = X % Y;
      if (M == 0) break;
          X = Y;
          Y = M;
      }
        printf("두 정수의 최대공약수(GCD) : %d\n" , Y);
        printf("두 정수의 최소공배수(LCM) : %d\n" , (A*B) / Y);

      return 0;
    }
```

- **배열** : 한번의 선언으로 여러개의 메모리 공간 확보 , 동일한 자료형의 연속적 모임

=> 배열변수 : int num[3] --> 순서도 num(3)

1) 배열변수 선언문 : 자료형 변수명; , 자료형 배열명 갯수(정수상수)

```
int a[10];                      ※ 4byte * 10
double b[2];                    ※ 8byte * 2
char ch[5];                     ※ 1byte * 5
```

2) 배열의 초기화

- 배열 요소의 범위 : 0번지부터 시작 , 배열선언과 동시에 초기화 시 요소의 개수 생략가능

```
ex)
        int a[3] = {1,2,3};
        double b[2] = {1.1,2.2};
        char ch[4] = {'P','A','S','S'}

        int a[] = {1,2,3};
        double b[] = {1.1,2.2};
        char ch[] = {'P','A','S','S'}
```

- **1차원 배열**

```
#include <stdio.h>

    void main()
    {
      int pil[5]; float avg;
      int i; total=0;
    for(i=0; i<5; i++)
    {
      scanf("%d",pil[i]):
```

```
    }
      avg = (pil[0]+pil[1]+pil[2]+pil[3]+pil[4]) / 5.0
      printf("정보처리 필기 평균점수 : %.2f점\n" , avg)
  }

    #include <stdio.h>

    void main()
    {
      int i;
      char ch[4] = {'P','A','S','S'}; --> char str[5] = {"PASS"}; 로 표현 가능
    for (i=0; i<4; i++)
    {
      printf("%c", ch[i]);
    }
      printf("\n");
  }
```

- **문자열 배열**

```
#include <stdio.h>

    void main()
    {
      int i;
      char  ch[4] = {'P','A','S','S'}           ※ '문자상수'들
      char  str[5] = "PASS";                    ※ "문자열상수" 1개 * 1개일 경우 {} 생략
                                                ※ 문자열의 끝에 마지막을 나타내는 '\0' 표기

        printf("문자배열의 크기 : %d바이트\n", sizeof(ch));
        printf("문자열배열의 크기 : %d바이트\n", sizeof(str));
    }


        ex) printf("%s\n" , str);               ※ printf("%s\n", str[0]);
```

- **2차원 배열**

```
#include <stdio.h>

    void main()
    {
      int i, j, sub_total;
      int s[3][2] = {{10,20},{30,40},{50,60}};      ※ 3행 2열 *행렬구조(표) -> 물리적 구조

    for(i=0; i<3; i++) {
      sub_total = 0;

    for(j=0; j<2; j++);{
      sub_total += s[i][j];
    }
      printf("%d번 학생 총점 : %d\n" , i+1 , sub_total);
        }
  }
```

- **포인터** : 특정 데이터가 저장된 주소값을 저장하는 변수 (하나의 주소값은 1byte)
- **포인터(pointer) 변수** : 주소(포인터 값)를 다룰 수 있는 변수

=> &(주소연산자)

=> 포인터 변수 선언문

```
int *p1;                          ※ 정수변수 주소 대입
int* p2;


p = &num;                         ※ 포인터자료형 변수 대입 , & : 주소연산자
                                  ※ *num 값의 주소위치가 대입

    #include <stdio.h>

    void main()
    {
      int num;
      int* p1;
      num = 100;
      p1 = &num;                                    ※ num변수의 주소위치 대입
        printf("일반변수 접근 : %d\n" , num);
        printf("포인터변수 접근 : %d\n" , *p1);       ※ 주소위치 간접 접근 후 내용처리
    }

1. 포인터변수와 배열

    #include <stdio.h>

    void main()
    {
      int i
      int A[5] = {10,20,30,40,50};
      int* p;
      p = A;                       ※ p = &A[0]; 배열명은 배열의 첫요소의 주소!!

    for(i=0; i<5; i++)
    {
      print("%5d" , *(p+i);       ※ printf("%5d" , A[i]);
    }
  }

2. 포인터변수와 연산

    #include <stdio.h>

    void main()
    {
      int NUM = 98;
      int* ptr;
      ptr = &NUM;
      NUM = NUM + 1;                  ※ ++NUM , NUM++
        printf("%d\n" , num);
      *ptr = *ptr + 1;                ※ ptr 위치의 내용(값)
```

```
        printf("%d\n" , *ptr);
    }

   -. 근사값 : 배열 A(100), 절대값이 500이하 중에서 정수 33에 가장 가까운 근사값

    #include <stdio.h>

    int main()
    {
      int A[10] = {131, 450, -100, 150, 50, -10, 0, 10, 32, 1};
      int MinCha = 533;                    ※ 거리_최소값의 초기화
      int N = 0,
      int Ans = N;                         ※ 근사값의 위치 (~번째)
      int Cha =0;                          ※ 거리_양수

    do {

    // 33기준으로 한 거리_양수
    if (A[N] >= 33)
      Cha = A[N] - 33;
    else
      Cha = 33 - A[N];

    // 거리의 최소값
    if (Cha < MinCha) {                    ※ if(MinCha > Cha)
                                           ※ 최소값 갱신
      MinCha = Cha;
      Ans = N;
    }
      N++;
    } while (N < 10);
        printf("33에 가장 가까운 값 : %d\n" , A[Ans]);
        printf("33에 가장 가까운 값의 첨자(위치) : %d\n" , Ans
        printf("33에 가장 가까운 값의 배열내의 위치 : %dth\n" , ANS+1);
  }
```

- **행 우선 / 열 우선 배열 채우기**

=> 2차원 배열(행열)에 값 채우기
=> 행 첨자(인덱스) : 0부터 ~
=> 열 첨자(인덱스) : 0부터 ~
=> A[행][열] : 값;

```
1. 행 우선 정사각형 배열 채우기

    1  2  3  4  5
    6  7  8  9 10
    11 12 13 14 15
    16 17 18 19 20
    21 22 23 24 25

    #include <stdio.h>

    int main()
    {
```

```
      int A[5][5];                  ※ int A[행수][열수]
      int i;                        ※ 행위치(첨자), 외부반복
      int j;                        ※ 열위치(첨자), 내부반복
      int value = 1;                ※ 1~25, +1
                                    ※ A[행][열] = 값;
    for(i=0; i <= 4; i++)           ※ 외부반복
    {
      for(j=0; j <= 4; j++)         ※ 내부반복
      {
        A[i][j] = value;            ※ A[행][열] = 값;
        value++;
      }
    }
                                    ※ A[행][열]의 값 출력
    for(i=0; i <= 4; i++)           ※ 외부반복
    {
      for(j=0; j <= 4; j++)         ※ 내부반복
      {
        printf("%3d", A[i][j]);
      }
        printf("\n");               ※ 줄변경 출력
    }
    return 0;
  }


== 연습란 ==
 행(i)   열(j)
-----------------
 0 --> 0,1,2,3,4
 1 --> 0,1,2,3,4
 2 --> 0,1,2,3,4
 3 --> 0,1,2,3,4
 4 --> 0,1,2,3,4

2. 열 우선 정사각형 배열 채우기

    1  6  11  16  21
    2  7  12  17  22
    3  8  13  18  23
    4  9  14  19  24
    5 10  15  20  25

== 연습란 ==
열(C) 행(R)
0 --> 0,1,2,3,4
1 --> 0,1,2,3,4
2 --> 0,1,2,3,4
3 --> 0,1,2,3,4
4 --> 0,1,2,3,4

    #include <math.h>

    int main()
    {
      int A[5][5];                  ※ 배열변수 선언문;
      int R, C;
      int V = 0;                    ※ 1~25, +1
      C = 0;
```

```
    do {
        int R = 0;
    do {
        V++;
        A[R][C] = V;
        R++;
      } while (R <= 4);               ※ 내부반복(변화)
        C++
    } while (C <= 4);                 ※ 외부반복(고정)
                                      ※ A[행][열]의 값 출력
    for (R = 0; R < 5; R++) {
      for(C = 0; C < 5; C++){
        printf("%d\t" , A[R][C]);     ※ t: tab간격 띄우기
        printf("\n");
      }
    }
    return 0;
  }
```

3. 행 우선 삼각형 배열 채우기

```
    1
    2  3
    4  5  6
    7  8  9  10
    11 12 13 14 15
```

== 연습란 ==

```
행(i) 행(j)
0 --> 0          (0~0, +1)
1 --> 0,1        (0~1, +1)
2 --> 0,1,2      (0~2, +1)
3 --> 0,1,2,3    (0~3, +1)
4 --> 0,1,2,3,4  (0~4, +1)
```

```
    #include <stdio.h>

    int main()
    {
      int A[5][5] = {0};                  ※ int A[행수][열수]
      int i;                              ※ 행위치(첨자), 외부반복
      int j;                              ※ 열위치(첨자), 내부반복
      int value = 1;                      ※ 1~25, +1
                                          ※ A[행][열] = 값;
    for(i=0; i <= 4; i++)                 ※ 외부반복
    {
      for(j=0; j <= i; j++)               ※ 내부반복
      {
        A[i][j] = value;                  ※ A[행][열] = 값;
        value++;
      }
    }
                                          ※ A[행][열]의 값 출력
    for(i=0; i <= 4; i++)                 ※ 외부반복
    {
      for(j=0; j <= 4; j++)               ※ 내부반복
      {
        printf("%3d", A[i][j]);
      }
```

```
        printf("\n");                    ※ 줄변경 출력
    }
    return 0;
  }

== 연습란 ==
행(R) 행(C)        R~4, +1
0 --> 0,1,2,3,4   (0~4, +1)
1 --> 1,2,3,4     (1~4, +1)
2 --> 2,3,4       (2~4, +1)
3 --> 3,4         (3~4, +1)
4 --> 4           (4~4, +1)

    #include <stdio.h>

    int main()
    {
      int A[5][5] = {0};                 ※ 배열요소 0으로 초기화
      int R, C;
      int V=1;                           ※ 1~15, +1
      R = 0;
    do{
        C = R;
    do{
        A[R][C] = V;
        V++;
        C++
      } while(C <= 4);                   ※ 내부반복(변화)
        R++;
    } while(R <= 4);                     ※ 외부반복(고정)
                                         ※ A[행][열]의 값 출력
    for(R = 0; R < 5; R++)
    {
      for(C = 0; C < 5; C++)
      {
        printf("%d\t" , A[R][C]);
    }
        printf("\n");
    }
        return 0;
  }

4-1. 석차 구하기 (동점 인정)

    #include <stdio.h>

    int main()
    {
      int A[25][2]; int R[4];                ※ 석차를 저장하는 배열

    for(int i = 0; i < 25; i++)
      A[i][1] = (rand()%300)+1;              ※ 매출실적 난수 발생

    for(int i = 0; i < 25; i++) {
      A[i][0] = i + 1;                       ※ 대리점의 이름 정의
      R[i] = 0;
    }

    for(int i = 0; i < 25; i++) {
```

```
      for(int j = 0; j < 25; j++) {
        if (A[i][1] <= A[j][1])
          R[i]++;
      }
          printf("%d(%d)\t : %d\n" , A[i][0], A[i][1], R[i]);
          ※ 대리점의 이름과 해당 석차 출력결과
    }
      printf("\n");
  }

    #include <stdio.h>
    #include <stdlib.h>                    ※ rand()함수 : 난수 발생

    int main()
    {
      int A[25][2];                        ※ 0열 : 대리점번호, 1열 : 매출실적
      int R[25];                           ※ 등수
      int i, j;

    for (i=0; i <= 24; i++)
    {
      A[i][0] = i + 1;                     ※ 대리점번호 저장
      A[i][1] = (rand()%300) + 1;          ※ 대리점매출실적 저장
      R[i] = 0;                            ※ 석차 초기화
    }

    for(i = 0; i <= 24; i++)               ※ 외부반복
    {
      for(j = 0; j<25; j++)                ※ 내부반복
      {
        if(A[i][1] <= A[j][1])             ※ 동점인정의 경우
        {
          R[i]++; // R[i] = R[i] + 1;
        }
      }
          printf("번호 : %2d" "실적 : %5d" "석차 : %2d\n" , A[i][0], R[i]);
    }
    return 0;
  }
```

4-2. 석차 구하기 (동점 인정 X)

```
    #include <stdio.h>

    int main()
    {
      int KUK[3], MAT[3], SUM[3];
      int RANK;
      int T;
      int K;
      int H;

    for(T = 0; T < 3; T++) {
      printf("%d번학생 정수입력(국어, 수학) : ", T);
      scanf("%d %d" , &KUK[T], &MAT[T]);
      SUM[T] = KUK[T] + MAT[T];
    }

    for (K = 0; K < 3; K++) {
```

```
      RANK = 1;

      for(H = 0; H < 3; H++) {
        if(SUM[K] < SUM[H])
          RANK = RANK + 1;
      }

      printf("%d번학생 : %5d %5d --> 총점 : %5d 순위 : %5d등\n" , K, KUK[K], MAT[K], SUM[K],
 RANK);
    }
    return 0;
  }
```

4-3. 3명의 학생을 대상으로 한 석차

```
    #include <stdio.h>

    int main()
    {
      int KUK[3], MAT[3], SUM[3];
      int RANK;                              // 석차
      int T, K, H;

    for(T=0; T<80; T++)
    {
      printf("%d번학생 정수입력(국어, 수학) : ", T);
      scanf("%d %d", &KUK[T], &MAT[T]);
      SUM[T] = KUM[T] + [T];
    }

    for (K = 0; K < 3; K++)
    {
      RANK = 1;                              // 각 학생은 1등으로 초기화

      for(H = 0; H < 3; H++)
      {
        if(SUM[K] < SUM[H])
        {
          RANK++; // RANK = RANK + 1
        }
      }

      printf("%d번학생 : %5d %5d --> 총점 : %5d 순위 : %5등\n" , K, KUK[K], MAT[K], SUM[K], R
ANK)
    }
    return 0;
  }
```

5. 선택정렬과 버블정렬

5-1. 선택정렬 : 학생 100명의 영어 성적을 오름차순으로 선택 정렬

```
    #include <stdio.h>

    int main()
    {
      int E[5]={95, 75, 85, 100, 50};
      int i = 0; int Temp;
```

```
    do{ int j = i + 1;

      do{
        if(E[i] > E[j] {
        Temp = E[i];
        E[i] = E[j];
        E[j] = Temp;
        }
        J++;
      } while (j<5);
        i++;
    } while (i<4);

    for (int a = 0; a < 5 a++){                    ※ 배열 E[] 출력결과
      printf("%d\t" ,  E[a]);
      printf("\n");
    }
    return 0;
  }

    #include <stdio.h>

    int main()
    {
      int E[5] = {95, 75, 85, 100, 50};
      int i, j, a;
      int Temp;                                    ※ 교환 임시변수
                                                   ※ 교환법 : 선택정렬(오름차순)
      i = 0;
    do{
      j = i + 1;
    do {
        if(E[i] > E[j])                            ※ 오름차순 (작->큰)
        {                                          ※ SWAP(교환) 알고리즘
          Temp = E[i];
          E[i] = E[j];
          E[j] = Temp;
        }
          j++
      } while(j<5)                          ※ 내부반복(변화) : 각 단계에서의 비교횟수
          i++;

      for(a = 0; a < 5; a++)
      {
        printf("%d\t" , E[a]);
      }
    } while(i<4);                           ※ 외부반복(고정) : 총단계(회전)수(4단계)

    for(a = 0; a < 5; a++)
    {
      printf("%d\t" , E[a]);
    }
      printf("\n");
    return 0;
  }

== 선택정렬 ==
i(4단계)     j(비교횟수)     (i+1~4,+1)
-------------
```

```
0 0:1 0:2 0:3 0:4 (1~4, +1)
1 1:2 1:3 1:4 (2~4, +1)
2 2:3 2:4 (3~4, +1)
3 3:4 (4~4, +1)

== 버블정렬 ==
i(4단계)     j(비교횟수)     (0~(3-i),+1)
-------------
0 0:1 1:2 2:3 3:4 (1~4, +1)
1 0:1 1:2 2:3 (2~4, +1)
2 0:1 1:2 (3~4, +1)
3 0:1 (4~4, +1)
```

5-2. 버블정렬

```
    #include <stdio.h>

    int main()
    {
      int E[5] = {95, 75, 85, 100, 50};
      int i, j, a;
      int Temp;                    ※ 교환 임시변수
                                   ※ 교환법 : 버블정렬(오름차순)

      i = 0;
    do{
          j = 0;
      do {
          if(E[j] > E[j+1])          ※ 오름차순 (작->큰)
          {                          ※ SWAP(교환) 알고리즘
            Temp = E[j];
            E[j] = E[j+1];
            E[j+1] = Temp;
          }
            j++
        } while(j<4-i)               ※ 내부반복(변화) : 각 단계에서의 비교횟수
            i++;
      for (a = 0; a < 5; a++)
      {
        printf("%d\t" , E[a]);
      }
    } while(i<4);                    ※ 외부반복(고정) : 총단계(회전)수(4단계)

    for(a = 0; a < 5; a++)
    {
      printf("%d\t" , E[a]);
    }
      printf("\n");
    return 0;
  }
```

-. 삽입정렬

```
    #include <stdio.h>

    int main()
    {
      int E[5] = {95, 75, 85, 100, 50};
      int i; int j,
```

```
       int k;
       int KEY;                     ※ 삽입값(대상)
                                    ※ 삽입법 : 삽입정렬(오름차순)

     for (i = 1; i < 5; i++) {      ※ 외부반복(고정), 총 단계수
         KEY = E[i];                ※ 삽입값 지정
       for (j = i-1; j >= 0; j--) { ※ 내부반복(변화), 비교횟수
          if(E[j] <= KEY) break;
           E[j+1] = E[j];
        }
          E[j+1] = KEY;
    }

    for (int a = 0; a < 5; a++){
      printf("%d\t" , E[a]);
      printf("\n");
    }
    return 0;
  }
```

6. 삽입정렬

```
    #include <stdio.c>

    int main()
    {
      int E[5] = {95 , 75 , 85 , 100 , 50}
      int i, j;
      int k;
      int KEY;                    ※ 삽입값(대상)
                                  ※ 삽입법 : 삽입정렬(오름차순)

    for (i = 1; i < 5; i++)       ※ 외부반복(고정), 총 단계수
    {
        KEY = E[i]                ※ 삽입값 지정
      for (j = i-1; j >=; j--)    ※ 내부반복(변화), 비교횟수
      {
        if(E[j] <= KEY            ※ 오름차순(작->큰)
        {
          break;                  ※ 배열(정열 O)에서 삽입위치(j+1번째) 찾음
        }
          E[j+1] = E[j];
      }
      E[j+1] = KEY;
    }

    for(k = 0; k < 5; k++)
    {
      printf("%d\t" ,  E[k]);
    }
      printf("\n");               ※ 줄 변경 출력
    return 0
  }
```

- **포인터변수 예시**

```
int main()
{
int num1 = 100, num2 = 100;
int * pnum;

    pnum = &num1;
    (*pnum) += 30;

    pnum = &num2;
    (*pnum) -= 30;

    printf("num1 : %d, num2 : %d \n", num1, num2);
    return 0;
    {
```