

경남교육 2008-015



정보올림피아드 및 프로그래밍 교육 교재

2008

교재의 특징 및 활용 방법

정보올림피아드 및 프로그래밍 교육 교재는 정보올림피아드 지도교사와 교원컴퓨터 프로그래밍경진대회에 참가하기 위하여 준비하는 교원에게 필요한 정보와 지도방법을 제시한 자료로 본 교재의 특징과 활용 방법은 다음과 같다.

1. 교재의 특징

첫째, Visual C와 Visual Basic을 배울 수 있는 프로그래밍 문법을 I 장과 II 장에 배치하였다.

둘째, 프로그래밍 문법을 배우면서 코딩을 하도록 기술하였으며 문법에서 출제된 기출문제를 추가하였다.

셋째, III장에서는 교육교재의 연습문제를 각 개념의 바로 다음에 배치하였고 기존 기출 문제를 응용한 연습문제를 별도로 제작하였다.

마지막으로, IV장과 V장에 알고리즘과 알고리즘 설계법을 배치하여 더 많은 공부를 원하는 학습자의 만족도를 높이도록 노력하였다.

2. 교재의 활용

이미 프로그래밍 언어를 선택하였고 문법은 어느 정도 숙달했다고 생각하는 학습자는 III장의 이산수학부터 공부하기를 권한다. 수학적 개념을 어느 정도 파악하고 실제 알고리즘에 대해 더 많은 공부를 원하는 학습자는 IV장부터 공부하면 된다.

이 책을 처음 접하는 학습자는 먼저 자신에게 맞는 프로그래밍 언어를 선택하여야 한다. I 장과 II장을 읽어보고 코딩을 해 보기를 권한다. Visual C와 Visual Basic은 서로 장단점이 있는 언어이다. 초보자용으로는 Visual Basic이 적당하고 Visual Basic은 문자열 조작이 상당히 편리하다. Visual Basic으로도 알고리즘을 구현할 수 있으나 전문 서적은 주로 C 프로그램을 기반으로 기술되었기 때문에 프로그래밍 실력을 향상

시키고자 할 때 애로점을 느끼게 될 것이다. Visual C를 선택한다면 www.dovelet.com에서 자동채점을 해 볼 수 있다. 처음에 조금 힘들고 시간이 걸려도 계속해서 프로그래밍 공부를 하려는 학습자에게는 Visual C를 권한다. 이제 자신에게 맞는 언어를 선택했다고 가정하고 교재를 보는 방법에 대해서 각 단원별로 언급하겠다.

I 장은 Visual C를 선택한 학습자를 위해 기본 문법을 설명한다. 그리고 세세한 과정과 불필요한 설명은 생략했기 때문에 부드럽게 학습할 수 있을 것이다. 기출문제는 꼭 풀어보기를 권한다.

II장은 Visual Basic을 선택한 학습자를 위해 기본 문법을 설명한다. 기술 순서는 I 장과 거의 비슷하지만 두 언어의 차이가 있으므로 그 점을 염두에 두기를 바란다.

III장의 이산수학은 출제빈도가 높은 6개의 단원(수의 표현, 계수, 그래프, 트리, 스택과 큐)으로 편성하였다. 계수는 기존의 순열과 조합에 비둘기 집의 원리를 추가하였다. 스택과 큐에서 예선 문제가 출제되고 있으므로 내용 설명과 기출 문제 및 구현까지 포함하였다.

IV장의 알고리즘에서는 각종 경시대회에 주로 사용되는 알고리즘들의 원리를 설명하였다. 그리고 직접 활용할 수 있도록 함수 형태의 코딩을 수록하였으므로 특정 알고리즘이 필요한 문제에 바로 적용하거나 약간의 수정을 통하여 이용할 수 있도록 구성하였다.

V장은 교원프로그래밍경진대회의 4, 5번 문제를 풀거나, 한국정보올림피아드에 출전할 학생들을 지도하고자 하는 선생님들에게 적당하다. 이 장에서는 특정 알고리즘을 이용하는 것이 아니라 문제에 적합한 알고리즘을 직접 설계하는 방법에 대해서 다루고 있다. 그리디, 동적계획, 분할정복, 백트래킹과 같이 알고리즘을 직접 만드는 기법들을 다루기 때문에 문제에 적합한 알고리즘을 효율적으로 설계 할 수 있을 것이다. 그리고 동일한 문제를 이용해서 각 알고리즘 설계법으로 알고리즘을 만들어 비교하고 있으므로 각 알고리즘이 어떤 문제에 효율적으로 적용될 수 있는지도 비교해 볼 수 있다. 마지막으로 교원프로그래밍경진대회의 기출문제 3~5번 문제의 풀이를 다루고 있다.



차 례

I. Visual C++	3
1. Visual C++ 시작하기	5
2. 자료형과 연산자	6
가. 상수(Constant) 7 / 나. 변수 7 / 다. 연산자 10	
3. 전처리와 표준 입출력문	17
가. 전처리기 17 / 나. 표준 입출력문 17	
4. 파일 입출력	22
가. 텍스트 문서 만들기 22 / 나. 파일 입출력 22	
5. 디버깅	27
가. 디버깅 모드 실행하기 27	
6. 제어문	30
가. 선택문 30 / 나. 반복문 / 32	
7. 함수	43
가. 함수의 기본적인 표현 구조 43 / 나. 인자의 전달 방식 44	
다. 변수의 범위 (지역변수와 전역 변수) 46 / 라. 되부름 함수 47	
8. 배열	54
가. 1차원 배열 54 / 나. 2차원 배열 56	
9. 포인터	68
가. 주소 연산자(&) 68 / 나. 간접 번지 연산자 (*) 68	
II. Visual Basic	77
1. Visual Basic 6.0 시작하기	79
가. Visual Basic 6.0 실행법 79	

2. 자료형과 연산자	83
가. 상수 83 / 나. 변수 83 / 다. 사용자 정의 자료형 88 / 라. 연산자 89	
3. 파일 입출력	94
가. 텍스트 문서 만들기 94	
4. 디버깅	99
가. 중단점 이용하기 99 / 나. 단계별 실행하기 100 / 다. 조사식 창 활용하기 100	
라. 디버깅 상태에서 특정 변수의 값 검사 101	
5. 제어문	102
가. 선택문 102 / 나. 반복문 106	
6. 프로시저	120
가. 함수(Function) 프로시저 120 / 나. 서브 프로시저 122	
다. 내장 함수 123 / 라. 변수의 범위 (지역변수와 전역변수) 125	
마. 되부름 함수 126	
7. 배열	134
가. 1차원 배열 134 / 나. 다차원 배열 135	

III. 이산수학 147

1. 수의 표현	149
가. 수의 표현 141 / 나. 진법의 변환 150 / 다. 보수 156	
2. 계수	160
가. 순열 160 / 나. 조합 163 / 다. 비둘기 집의 원리 166	
3. 관계	172
가. 관계 172 / 나. 관계의 표현 방법 173 / 다. 관계의 성질 175	
라. 동치관계 180 / 마. 부분순서관계 182	
4. 그래프	191
가. 그래프 191 / 나. 그래프의 종류 201 / 다. 그래프 탐색 210	
라. 최단경로 알고리즘 215	

5. 트리	222
가. 트리	222
나. 이진트리	226
다. 트리 순회	231
라. 수식트리	238
마. 최소비용신장트리	241
바. 이진탐색트리	246
사. 힙	252
야. 헤프만 코드	257
6. 스택	266
가. 스택	266
나. 스택의 구현	267
7. 큐	279
가. 큐	279
나. 큐의 구현	280

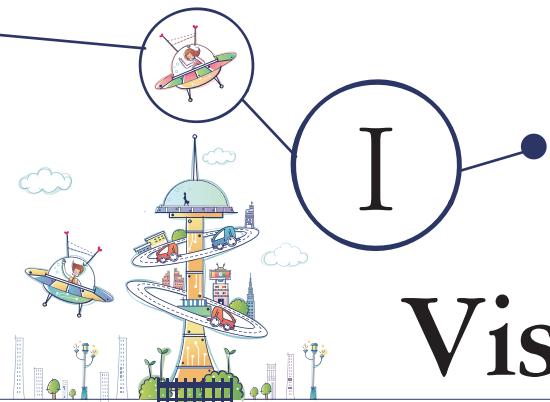
IV. 알고리즘 289

1. 알고리즘	291
가. 알고리즘의 정의	291
나. 알고리즘의 예	292
다. 알고리즘의 분석	300
2. 검색 알고리즘	308
가. 검색의 개요	308
나. 순차 검색	308
다. 이분 검색	310
라. 해시	313
3. 정렬 알고리즘	318
가. 선택정렬	318
나. 퀵정렬	321
4. 트리 관련 알고리즘	326
가. 이진트리의 표현	327
나. 이진트리의 순회	330
다. 이진 트리의 응용	333
5. 그래프 관련 알고리즘	350
가. 그래프의 표현	350
나. 그래프의 탐색	353
다. 그래프의 활용	361

V. 알고리즘 설계법	377
1. 알고리즘 설계론	379
2. 그리디(Greedy) 설계 기법	380
가. 그리디 설계기법이란 380	
3. 백트래킹(Backtracking) 설계 기법	387
가. 백트래킹 설계 기법이란 387 / 나. 백트래킹 설계 기법의 활용 395	
4. 분할정복(Divide And Conquer) 설계 기법	401
가. 분할정복 설계 기법이란 401 / 나. 분할정복 설계 기법의 활용 409	
5. 동적계획법(Dynamic Programming)	416
가. 동적계획법이란 416 / 나. 동적계획법의 활용 420	
6. 교원컴퓨터프로그래밍경진대회 본선 기출 문제 풀이	424
가. 제1회 424 / 나. 제2회 435 / 다. 제3회 444 / 라. 제4회 454	



- 정보올림피아드 및 프로그래밍 교육 교재



Visual C++

제 I 장에서는 Visual C++ 프로그래밍 기초에 관련된 내용을 살펴본다. 정보올림피아드와 교원컴퓨터프로그래밍 경진대회에서 가장 많이 사용하는 Visual C++ 6.0의 툴 사용법과 C언어의 기본 문법, 기출문제 풀이 등을 수록하였다.

1. Visual C++ 시작하기

C 언어는 프로그래밍 언어를 공부하는 사람이라면 누구나 배우게 되는 언어이다. 교육용으로 아주 강력한 프로그램이다. 또한 다양한 기초 교육 교재, 자료구조와 알고리즘 등의 복잡한 내용까지 다루는 고급 교재 등 다른 언어에 비해 훨씬 많은 참고물이 제공되고 있다. 정보올림피아드나 각종 대회에서도 C언어를 사용하는 사람이 가장 많다. 여기서는 Visual C++ 6.0을 이용하여 실습을 할 것이며 가능한 이론 중심에서 벗어나 실습 문제를 많이 다룰 것이다. 먼저 Visual C++ 6.0 프로그램의 기본적인 실행 방법에 대해서 살펴보도록 하자.

가. 일반적인 사용법

- ① [File] → [New] 메뉴를 실행시킨다.
- ② New 대화상자가 뜨면 [Projects 탭]을 선택하고, [Win32 Console Application]을 선택한다.
- ③ New 대화상자에서 [Location : 의] 버튼을 누르면 [Choose Directory 대화상자]가 새로 나타난다. 프로젝트를 저장할 폴더를 선택한 후 OK 버튼을 누른다.
- ④ New 대화상자에서 [Project name :]에서 생성할 프로젝트의 이름을 입력한 후 OK 버튼을 누른다.
- ⑤ 만약 [Win32 Console Application - Step 1 of 1 대화상자]가 나타나면 [An empty project.]를 선택한 후 Finish 버튼을 누른다.
- ⑥ 이번에는 [New Project Information 대화상자]가 나타나는데 바로 OK 버튼을 누른다.
- ⑦ [File] → [New] 메뉴를 실행한다. New 대화상자의 [Files 탭]에서 [C++ Source File]를 선택, 오른쪽 편의 [Add to project]에 체크, [Location :]에서 소스 파일의 저장 위치를 ③번과 동일하게 설정한다.
- ⑧ New 대화상자에서 마지막으로 [File 이름 입력란]에 소스 파일의 이름을 적어주고 OK 버튼을 누르면 된다. 이것으로 Visual C++에서 프로그래밍을 하기 위한 사전 작업은 모두 끝이 나게 되고 코딩을 시작하면 된다.

- ⑨ 소스를 작성하고, [File] → [Save] 메뉴를 실행하여 저장하면 된다. 저장할 때 소스는 확장자가 반드시 *.c이나 *.cpp 형식이어야 한다. 확장자 *.c는 컴파일 할 때, c 문법으로 컴파일이 되고, *.cpp는 c++ 문법으로 컴파일 된다. 대회에서는 확장자 *.cpp를 사용할 것을 권장한다.
- ⑩ [Build] → [Compile] 메뉴를 실행시킨다. (도구나 단축키 **Ctrl+F7** 이용) 컴파일러가 소스의 문법적 오류나 에러를 아웃풋 영역에 표시해 준다.
- ⑪ 오류가 있을 경우 컴파일한 결과에서 에러난 부분을 더블클릭하면 소스의 해당되는 줄에 커서가 바로 옮겨진다. 오류를 수정한 뒤 다시 컴파일한다.
- ⑫ [Build] → [Execute] 메뉴를 실행하여 실행 파일의 생성한다.
- ⑬ 실행 화면이 나타날 것이다. 아무키나 누르면 실행 화면이 없어진다.
- ⑭ 탐색기로 소스를 저장한 폴더 안의 Debug 폴더를 탐색해 보면 실행 파일 들어 있을 것이다. 보통 제출해야 하는 파일은 파일 입출력을 할 경우 소스파일, 실행파일, 입출력파일 등 4개의 파일이며 한 문제당 하나의 폴더에 넣어서 제출해야 한다.
- ⑮ [File] → [Close Workspace] 메뉴를 실행하여 작업그룹을 닫은 다음 새로운 프로그램을 하면 된다.
- * 작업 도중에 실수로 Visual C++ 프로그램을 닫았다면 [File] → [Open Workspace] 메뉴를 실행하여 워크스페이스를 불러와 소스를 열어야 한다.
- * 작업 도중에 워크스페이스는 닫지 않고 프로그래밍을 하던 소스창만 닫았다면 화면의 왼쪽의 워크스페이스 작업영역에서 [FileView]를 클릭하여 워크스페이스 파일 이름을 누르면 3개 폴더가 나온다. [Source Files]를 더블 클릭하면 소스 창이 나타날 것이다.

2. 자료형과 연산자

자료형은 데이터가 가질 수 있는 형태를 이야기 하는 것으로, 여기에서는 상수, 변수에 대해 다양한 자료형들을 알아보고, 연산자의 종류와 사용법에 대해서도 살펴보도록 하자.

가. 상수(Constant)

한번 정의되면 변하지 않는 값을 **상수**라 한다. 상수의 종류에 대해 알아보자.

1) 정수형(Integer) 상수

소수점이 없는 음의 정수, 0, 양의 정수를 말한다. 종류로는 8진 상수(0으로 시작), 10진 상수(0이 아닌 수로 시작), 16진 상수(0x로 시작)와 long형 상수(L, 1문자로 끝남)가 있다.

2) 실수형 상수

소수점이 존재하는 실수(Floating-point, 부동소수점)를 말한다. 종류로는 10진 실수(200.35)와 지수형 실수(영문자 E 또는 e 문자를 포함, 0.2E10)가 있다.

3) 문자 상수

문자(Character)를 표현하는 것이다. 작은 따옴표(' ') 안에 하나의 문자를 넣어 표현하거나('h'), 큰 따옴표(" ") 안에 1개 이상의 문자를 넣어 문자열을 표현한다.

나. 변수

변수(Variable)는 프로그램이 실행될 때 어떤 값을 저장하기 위한 기억 장소를 말하며, 프로그램 실행 과정에서 값을 변경할 수 있다.

1) 변수의 선언 형식

변수를 선언하는 형식은 다음과 같다.

● 형식 : [기억클래스] [자료형] 변수명들 ;

- ① **기억클래스** : 변수가 어디에 기억장소를 잡을 것인가와 관련된 부분으로, auto, static, register와 extern이 있다. auto는 변수가 선언될 시점부터 선언된 함수 블록이 끝날 때까지 변수가 살아 있도록 만들어 준다. static은 변수가 컴파일 할 때 기억 장소가 확보되어 프로그램이 종료될 때 까지 살아 있도록 만들어 준다.

- ② 자료형은 기본적으로 정수형 (integer), 실수형 (floating-point)과 문자형 (character)이 있다.

2) 기본 자료형

자료형의 종류를 <표 I .2.1>를 통해 살펴보자. 형식 지정 문자는 scanf()문과 printf문 등으로 입출력을 할 때 사용하는 것이므로 잘 알아두자.

<표 I .2.1> 자료형

구 분	종 류	형의 크기	표현 범위	형식 지정문자
정수형	short	2byte	-32,768~32,767	%d
	int	4byte	-2,147,483,648~2,147,483,647	%d
	long	4byte	-2,147,483,648~2,147,483,647	%ld
	unsigned	4byte	0~4,294,967,295	%u
실수형	float	4byte	3.4E-38~3.4E+38	%f
	double	8byte	1.7E-308~1.7E+308	%f
	long double	8byte	1.7E-308~1.7E+308	%lf
문자형	char	1byte	-128~127	%c
형 없음	void			

※ 주의할 점은 형 변환이 이루어지면 값이 절단되는 경우가 발생한다.(float 형의 값이 int형으로 변환되면 소수점 이하의 값은 절단하고 대입된다.)

다음은 소문자를 대문자로 바꾸는 프로그램이다.

```
#include <stdio.h>
void main( )
{
    char ch1 = 'd', ch2 ; // 문자형 변수 선언
    ch2 = ch1 + ('A' - 'a') ;
    printf(" ch2 = %d ch2 = %c ", ch2, ch2) ;
}
```

실행 결과	ch2 = 68 ch2 = D
-------	------------------

printf()문에서 %d는 10진수 형식으로, %c는 문자형으로 출력하라는 뜻이다. 문자형 변수에 저장된 값은 실제로 정수이기 때문에 연산도 가능하다. 영문자 'A'와 'a'는 아스키(ASCII)값이 각각 65와 97이다.



문제 I .2.1

다음 프로그램의 출력 결과는 무엇인가?(2005 초등)

```
#include <stdio.h>
void main( )
{
    printf("%d ", 10 - 3 * 2 ) ;
}
```

- ① 4 ② 6 ③ 8 ④ 10 ⑤ 14

<풀이> ①

3) 사용자 정의 자료형

기본 자료형 외에 사용자가 필요에 따라 정의하여 사용하는 사용자 정의 자료형이 있다. C언어에서는 구조체(structure), 공용체(union)와 열거형(enum) 등이 있다. 여기서는 많이 사용되는 구조체에 대해서만 살펴보자.

가) 구조체(Structure)

구조체는 서로 다른 자료형의 자료들을 하나의 집단으로 묶고 하나의 이름으로 관리하는 것을 말한다. 구조체는 사용자가 새로 정의한 자료형이다. 구조체 자료형의 구조체 변수를 선언하여 관리해야 할 자료들을 저장하고 처리하게 된다. 구조체 변수 안에는 필드(field)들을 가지게 되고 필드명에 접근할 때에는 구조체 연산자 '.'이나 '->' 연산자를 이용하게 된다. '.' 연산자는 왼쪽에 구조체명이 오고, '->' 연산자는 왼쪽에 구조체 포인터명이 온다. 구조체의 선언은 struct라는 예약어를 사용하여 다음과 같이 선언한다.

● 형식 :

```
struct 구조체명{
```

```
    항목들 ;
```

```
}
```

다음은 한국중학교 1학년 인적자료들을 입력받기 위해서 student라는 새로운 자료형(구조체)을 선언한 것이다.

```
struct student{
    int ban ;           // 반
    int num ;           // 번호
    char name[15] ;     // 이름, [ ]기호는 배열을 의미
    char add[40] ;      // 주소
    char tel[15] ;      // 전화번호
};
```

구조체가 선언되면 일종의 데이터 형만 선언된 것이므로 실제적으로 프로그램 상에서 사용할 변수를 선언해야 한다. 변수 선언은 다음과 같다.

다. 연산자

연산자는 자료 처리를 위한 수식들을 결합하여 연산 동작을 수행하도록 하는 기호를 말한다. 그리고 연산에 사용되는 자료값들을 피연산자(오퍼랜드)라 한다.

- ① 산술 연산자 : 사칙연산자(+, -, *, /)와 나머지(%) 연산자 등이 있다.
- ② 대입 연산자 (=) : 오른쪽의 계산 결과를 왼쪽의 변수에 대입(저장)하는 연산자이다.

〈표 I .2.2〉 대입 연산자와 복합 대입 연산자

연산자	사용법	같은 산술식	설명
=	a = b ;		오른쪽의 값을 왼쪽에 대입
+=	a += b ;	a = a + b ;	a와 b를 더한 결과를 왼쪽 a에 대입
그 외에도 -=, *=, /=, %=, <<=, >>, &=, ^=, = 등이 있다.			

③ 증감 연산자 (++ , --)

보다 간결하게 표현하고 실행 속도를 빠르게 하기 위해 사용하는 것으로 1씩 증가하는 연산자 ++와 1씩 감소하는 연산자 --가 있다. 예를 들어 a=5일 때, 증가연산자에 연산 결과를 알아보자.

〈표 I .2.3〉 증감 연산자

종류	사용법	b의 값	설명
증가연산자 (+ +)	b = ++a	6	a를 먼저 1만큼 증가한 후 a의 최종값을 b에 대입
	b = a++	5	a를 변수 b에 대입한 후 a를 1만큼 증가

다음은 증감 연산자에 대한 예제이다.

```
#include <stdio.h>
void main( )
{
    char k = 'N' ;                                // 'N'==78
    int i, j ;
    i = k ++ ;                                    // i ==78 , k == 79
    j = ++ k + ( -- i ) ;                         // j = 80+77 , i==77
    printf(" i = %c    j = %d \n", i--, j) ;
}
```

실행 결과	i = M j = 157
-------	---------------

④ 관계 연산자

2개의 피연산자에 대해 대소 관계를 나타내는 연산자이다. 관계의 결과값이 참인 경우엔 1, 거짓인 경우엔 0이 된다.

〈표 I .2.4〉 관계 연산자

종 류	사 용 법	의 미
==	a == b	a는 b와 같다
!=	a != b	a는 b와 다르다
그 외에도 <, >, <=, >= 등이 있다.		

⑤ 논리 연산자

2개의 피연산자에 대해 참인지 거짓인지 구분하는 연산자이다. 조건이 참인 경우엔 1, 거짓인 경우엔 0이 된다.

〈표 I .2.5〉 논리 연산자

종류	사용법	의 미	
&&	A && B	논리곱, AND	A와 B 모두가 참이면 결과가 참이 된다.
	A B	논리합, OR	A와 B 둘 중 한 개 이상 참이면 결과가 참
!	!A	논리부정, NOT	A의 부정, A의 값이 0이면 참

⑥ 비트 연산자

2진수 체계로 비트 단위로 논리곱, 논리합, 배타적 논리합과 보수 등을 표현하는 연산자이다.

〈표 I .2.6〉 비트 연산자

종류	사용법	의 미	
&	a & b	논리곱, AND	a와 b를 비트 단위별로 논리곱을 함
	a b	논리합, OR	a와 b를 비트 단위별로 논리합을 함
^	a ^ b	배타적 논리합, XOR	a와 b를 비트 단위별로 배타적 논리합을 함
~	~a	보수	a의 1의 보수를 취함(0은 1로, 1은 0으로)

⑦ 이동(shift, 쉬프트) 연산자

이동(shift) 연산자는 지정한 크기만큼 비트 단위로 좌·우로 이동시키는 연산자를 말한다. 그리고 가장 왼쪽의 첫 비트인 부호 비트는 움직이지 않으며, 왼쪽으로 이동할 때는 남은 빈칸은 0을 채우고 오른쪽으로 이동할 때는 남은 빈칸은 부호 비트와 같은 비트를 채운다

〈표 I .2.7〉 이동 연산자

종류	의미	사용 예	2진수 표현	연산 후 2진수 결과	10진수 결과
<<	비트를 좌측으로 이동	12<<2	00000000 00001100	00000000 00110000	48
		-12<<2	11111111 11110100	11111111 11010000	-48
>>	비트를 우측으로 이동	12>>2	00000000 00001100	00000000 00000011	3
		-12>>2	11111111 11110100	11111111 11111101	-3

⑧ 조건 연산자

조건식의 결과에 따라 참(True)인 경우에는 문장1을 수행하고 조건식이 거짓(False)인 경우에는 문장2를 수행하는 연산자로, 3개의 피연산자가 필요하므로 3항 연산자라고도 한다. 형식은 다음과 같다.

● 형식 : 조건식 ? 문장1 : 문장2

예) $(a > b) ? c = b : c = a ;$

⑨ 포인터 연산자 (&, *)

포인트 연산자에는 주소연산자(&)와 간접변지 연산자(*)가 있고 두 연산자 모두 피연산자가 1개 필요한 단항 연산자이다. 자세한 사항은 8. 포인터 부분에서 살펴보고 여기서는 간단히 언급만 하고 가자.

주소연산자(&)는 변수 바로 앞에 붙여서 변수가 메모리 내에 위치하고 있는 주소를 나타낸다. 형식은 다음과 같다.

● 형식 : &변수명

예) $\&ch$

`char ch='A'; int a=100;` 으로 선언되어 있을 때 메모리 일부분이 아래 그림과 같이 되어 있다면, $\&ch$ 는 1201번지를 나타낸다.

									메모리 주소(byte)
	A		1201		100				내용
	변수 ch		변수 pt		변수 a				변수명

간접변지 연산자(*)는 포인터 앞에 붙여서 포인터(pointer) 변수가 가리키는 메모리 주소에 기억되어 있는 내용을 나타낸 것이다. 위의 그림에서 포인터 변수 pt의 값이 1201이므로 $*pt$ 는 'A'를 나타내는 것이다.

포인트 변수 선언 형식은 다음과 같다.

● 형식 : 자료형 * 포인터 변수;

예) `char *pt, ch;`

포인트 변수 관련 예제를 살펴보자.

```

#include <stdio.h>
void main( )
{
    char *pt, ch;           // pt는 주소를 저장할 수 있는 포인트 변수
    ch = 'A';
    pt = &ch;              // 변수 ch의 주소를 pt에 대입
    printf(" ch = %c    pt = %c \n ", ch, *pt);
}

```

실행 결과

ch = A pt = A

⑩ 연산자의 종류와 우선순위

다양한 연산자의 종류와 우선순위를 〈표 I .2.8〉에서 살펴보도록 하자.

〈표 I .2.8〉 연산자와 우선순위

우선순위	연산자의 종류	연 산 자	결합 방법
↑↑↑↑↑↑↑↑↑↑	함수, 배열, 구조체	() [] ->	왼쪽 ⇒ 오른쪽
	단항 연산자	! ~ + - ++ -- & * (type) sizeof	왼쪽 ⇌ 오른쪽
	산술 연산자	* / %	왼쪽 ⇒ 오른쪽 왼쪽 ⇒ 오른쪽
		+ -	
	쉬프트 연산자	<< >>	
	비교 연산자	< <= > >= == !=	
	비트 연산자	& ^	
	논리 연산자	&&	
	조건 연산자	? :	
	대입 연산자	= *= /= %= += -= <<= >>= &= ^= =	왼쪽 ⇌ 오른쪽
	콤마 연산자	,	왼쪽 ⇒ 오른쪽



연습문제

1

수식 $\frac{100 \times 7^3}{4^3 - 3} = x$ 에서 x 의 결과를 출력하는 프로그램을 작성하시오.

2

출력 결과가 다른 하나는?(단, int a=4, b=2, c=0 ; 으로 선언됨)

- ① a || b | c ② false & b || c ③ a = 4 & c
- ④ b & c & a ⑤ ! (~ a || b)

3

다음 프로그램에서 모든 수행을 마친 후 a와 b변수에 기억되는 값은?
(strlen()함수는 문자열의 길이를 나타내는 함수임) (2004 초등)

```
#include <stdio.h>
#include <string.h>
void main ()
{
    char a[] = "I am a boy." ;
    printf("%d \n", strlen(a) ) ;
}
```

- ① 7 ② 8 ③ 9 ④ 10 ⑤ 11

4

출력 결과가 다른 하나는?(단, int a=10,b=15 으로 선언됨)

```
printf(" %d ", ( -- a + b -- & ~ a)) ;
```

- ① 8 ② 16 ③ 24 ④ 32 ⑤ 36



풀



7³ 지수 표현은 pow(7,3)으로 이용하여 쉽게 나타낼 수도 있다.
결과는 “ result = 2638.461”가 출력된다.

```
#include<stdio.h>
void main()
{
    float result ;
    result = (float) ( 100 * 7 * 7 * 7 ) / ( 4 * 4 - 3 ) ;
    printf(" result = %10.3f \n", result ) ;
}
```



①

①의 결과는 1이 출력되지만, 나머지 보기는 0을 출력한다.



⑤



②

우선 순위에 의해 &연산자가 마지막에 수행한다.

3. 전처리와 표준 입출력문

전처리기는 프로그램을 실제로 번역하기 전에 미리 헤더파일이나 매크로 등을 등록하거나 변경하는 것을 말한다. 표준 입출력은 입력은 키보드로 하고 출력은 모니터로 하는 것을 말한다. 전처리기와 표준 입출력문에 대해 알아보자.

가. 전처리기

전처리기는 컴파일러가 실제로 프로그램을 컴파일하기 전에 먼저 처리하는 것을 말한다.

① #include문

헤더(header) 파일을 삽입할 때 사용한다.

#include <stdio.h> //시스템이 제공하는 헤더 파일을 포함시킬 때

#include "head1.h" //사용자 정의 헤더 파일을 포함시킬 때

자주 사용되는 시스템 헤더 파일은 〈표 I .3.1〉과 같다.

〈표 I .3.1〉 자주 사용되는 헤더 파일

헤더 파일	내 용	함수 예	비 고
stdio.h	표준 입출력 함수	printf(), scanf()	c언어에선 반드시 포함
string.h	문자열 함수	strcpy(), strlen()	필요에 따라 포함
stdlib.h	기본 라이브러리 함수	qsort(), bsearch(), rand()	"
math.h	수학 함수	log(), pow()	"
time.h	시간 함수	time(), ctime()	"

② #define문 (매크로 선언)

#define문은 반복되는 상수나 특정한 문장 등을 프로그래머가 이해하기 쉬운 단어나 문장으로 재정의를 내리는 것이라고 할 수 있다. 이렇게 재정의 된 단어를 매크로(macro)라고 한다. 표현 형식은 다음과 같다.

● 형식 : #define [기호상수] [바꿀 상수 또는 문자열]

예) #define PHI 3.14

나. 표준 입출력문

입출력 함수의 종류는 표준 입출력 함수와 파일 입출력 함수가 있다. 표준 입력 함수는 입력은 키보드로 받고, 출력은 모니터로 한다는 것을 의미한다. 표준 입출력 함수를 사용하려면 #include<stdio.h>을 선언해야 한다. 파일 입출력 함수는 표준 입출력 장치 이외의 하드디스크 같은 저장장치에서 파일을 열어 입력과 출력을 하는 것이다. 먼저 표준 입출력의 종류부터 살펴보자

〈표 I .3.2〉 표준 입출력 함수

종 류	의 미	형 식	사용 예
scanf()	형식에 맞추어 문자열 입력	scanf("형식", 인수) ;	scanf("%c", &ch) ;
printf()	형식에 맞추어 문자열 출력	printf("형식", 인수) ;	printf("프린터") ;
getchar()	한 문자 입력	getchar() ;	ch = getchar() ;
putchar()	한 문자 출력	putchar(인수) ;	putchar(ch) ;
gets()	한 문자열 입력	gets(인수) ;	gets(st) ;
puts()	한 문자열 출력	puts(인수) ;	puts(st) ;

① printf()함수와 scanf()함수

printf()함수는 데이터를 원하는 형태로 형식(format)에 맞추어 모니터로 출력시키는 함수이다. scanf()함수는 키보드를 통해 형식에 맞게 데이터를 입력받는 함수이다. printf()함수와 scanf()함수의 형식은 다음과 같다.

● 형식 : printf("출력할 문장") ;

printf("형식지정-문자열", 인수들) ;

scanf("형식지정-문자열", 인수들) ;

예) int k ;

char a[10], *pt ; // 배열과 포인터 변수 선언

scanf("%d", &k) ; // k에 int형의 숫자를 입력받음

scanf("%s", a) ; // k는 배열명, 포인터로 문자열을 입력받음

scanf("%s", pt) ; // pt는 포인터, 문자열을 입력받음

printf("k = %d a = %s", k, a) ; // 정수와 문자열 출력

scanf() 함수에서 인수를 기술할 때에는 기억 장소를 나타내는 포인터를 사용해야 한다.

다음은 형식 지정 문자의 종류를 나타낸 표이다.

〈표 I .3.3〉 형식 지정 문자

형식지정문자 (변환 문자)	의 미	예	출력
%d	10진수로 입출력	printf("%d", 10) ;	10
%o	8진수로 입출력	printf("%o", 10) ;	12
%x	16진수로 입출력	printf("%x", 10) ;	a
%u	부호 없는 10진수로 입출력	printf("%u", 10) ;	10
%ld	long형의 10진수로 입출력	printf("%ld", 10) ;	10
%f	실수형으로 입출력	printf("%f", 10.05) ;	10.05
%c	문자형으로 입출력	printf("%c", 'A') ;	A
%s	문자열 형태로 입출력	printf("%s", "string") ;	string
%e	지수 형태로 입출력	printf("%e", 0.0000055) ;	5.500000e-6

형식지정 문자를 사용할 때 %nd와 같은 형태로 적음으로써 n개의 자리수 만큼 출력할 수 있다.

다음 예제를 통해 다양한 출력 방법을 살펴보도록 하자.

```
#include <stdio.h>
void main( )
{
    long lo = 4567L ;
    float fl = 124.68 ;
    printf("%10ld\n", lo) ;      //10자리 중에서 오른쪽에 배치
    printf("%10.1fn", fl) ;     //소수점 첫째자리 표현, 절단되는 부분은 반올림
    printf("%-10.3fn", fl) ;   // -는 왼쪽부터 배치
}
```

실행 결과	4567 124.7 124.680
-------	--------------------------



문제 I .3.1

두 숫자를 키보드로 입력받아 평균을 출력하는 프로그램이다. 출력 결과는?

(단, n, m은 각각 7, 8을 입력받았다.)

```
#include <stdio.h>
void main( )
{
    int n, m ;
    scanf(" %d %d", &n, &m) ;
    printf(" 합계 = %d \n", (n+m)/2 ) ;
}
```

- ① 7 ② 7.5 ③ 7.50 ④ 8 ⑤ 문법 에러

<풀이> ①

정수와 정수의 나눗셈의 결과는 정수가 나타난다. 따라서 소수점까지 만들려면 %f기호 사용, (float)(n+m)/2 형식으로 형변환을 시켜줘야 한다.

③
getchar() 함수와 putchar() 함수
키보드로 한 문자를 입력받는 함수이다.

● 형식 : getchar()

putchar(인수) // 인수는 문자형
예) char ch;
 ch=getchar();
 putchar(ch);

④
gets() 함수와 puts() 함수

gets() 함수는 키보드로 문자열을 입력받는 함수이며 인수는 배열 이름이거나 포인터 변수이어야 한다. puts() 함수는 모니터로 문자열을 출력하는 함수이며 배열 이름이나 포인터로 시작 주소를 받으면 그 주소의 내용부터 출력하여 NULL 문자가 나올 때까지 출력한다.

● 형식 : gets(인수)

puts(인수)
예) char ch[50];
 gets(ch);
 puts(ch);



연습문제



다음 프로그램의 출력 결과는?($a=246$, $b=135.790$ 으로 선언)

```
printf("%-5d", "%5.2f",a,b);
```

- ① [246], [135.790]
- ② [246], [135.79]
- ③ [246], [135.79]
- ④ [246], [135.79]
- ⑤ [246], [135.790]



다음 프로그램의 출력 결과는?

```
char *st="abcde";
printf("%3s",st);
```

- ① [abcde]
- ② [abc]
- ③ [cde]
- ④ [abcdef]
- ⑤ 문법 에러



풀

이



④ 숫자표현은 %nd에서 n이 음수이면 원쪽부터 자리수만큼 출력하고, 양수이면 오른쪽부터 자리수만큼 출력한다. 소수점 이하는 소수점 이하 자리수 표현을 나타낸다.



① 문자열 출력은 반드시 “\0” 기호가 있을 때까지 출력한다.

4. 파일 입출력

키보드나 모니터가 아닌 저장장치의 특정한 파일에서 입력 자료를 읽어오거나 출력 자료를 기록하는 일련의 과정들을 파일 입출력이라고 한다. 이용되는 파일에는 텍스트 파일(text file)과 이진 파일(binary file)이 있다. 프로그래밍 대회에선 반드시 입력파일에서 입력자료를 읽어오고 결과는 출력파일에 기록한다. 그래서 소스파일과 함께 입출력 파일도 결과물로 제출 해야 한다. 우선, 텍스트 문서를 만드는 방법부터 살펴보자.

가. 텍스트 문서 만들기

입출력파일은 메모장 등의 문서작성기로 파일을 생성할 수도 있고 Visual C++ 프로그램에서도 생성할 수 있다. 텍스트 파일을 생성한 다음에는 프로젝트에 등록하여 Visual C++ 프로그램에서 이동하는 것이 편리하다. 텍스트파일 생성 방법에 대해 살펴보자.

1) Visual C++ 프로그램에서 만들기

- ① [File] → [New] 메뉴를 실행시킨다.
- ② New대화상자의 [Text File]선택 후 [add to project]를 체크하고 저장한다.
- ③ 새 파일이 열리면 입력 파일의 내용을 입력하고 소스파일이 있는 위치에 저장한다.
- ④ 파일들의 이동은 컨트롤([Ctrl])키 + 탭([Tab])키를 이용하여 열어둔 파일들을 쉽게 이동할 수 있다.

나. 파일 입출력

원래 컴퓨터상에서는 모니터나 키보드 같은 표준 입출력도 파일로 처리를 하는데, 이것은 운영체제가 자동적으로 파일을 열어주고 닫아주기 때문에 우리는 별다른 작업 없이 처리를 하는 것이다. 하지만, 그 이외의 파일 입출력은 FILE문, fopen문, fclose문 등을 이용하여 사용자가 직접 파일을 연결하여 열어주고 닫아주는 작업과 fscanf문, fprintf문 등을 이용하여 입출력을 하는 작업을 직접 해주어야 한다. 앞으로 다룰 파일 입출력 함수의 종류에 대해 살펴보자.

〈표 I .4.1〉 파일 입출력 관련 함수들

종류	의 미	예	설 명
fopen()	파일을 열기	fp=fopen("pre.txt","r");	pre.txt 파일을 오픈함
fclose()	파일을 닫기	fclose(fp);	pre.txt 파일을 닫음
fscanf()	형식에 맞게 입력	fscanf(fp,"%d",&data);	fp가 연결된 파일에서 하나의 10진수를 입력받아 data 변수에 저장시킴
fprintf()	형식에 맞게 출력	fprintf(fp,"%d",data);	data값을 fp가 연결된 파일에 쓰기를 함.
fgetc()	한 문자를 입력	fgetc(fp,ch);	한 문자를 fp가 연결된 함수에서 입력받음
fputc()	한 문자를 출력	fputc(fp, ch);	한 문자를 fp가 연결된 파일에 쓰기를 함
fgets()	한 문자열을 입력	fgets(fp,st);	문자열을 fp와 연결된 파일에 입력받아 st에 저장
fputs()	한 문자열을 출력	fputs(fp,st);	문자열을 fp와 연결된 파일에서 쓰기를 함
feof()	파일 끝인지 검사	afeof(fp);	파일의 끝이면 -1을 a에 저장하고, 끝이 아니면 0을 a에 저장함

1) 파일 선언

파일에서 데이터를 입출력하려면 입출력 파일과 작성하는 프로그램을 연결하는 변수가 필요하다. 이 변수를 파일 포인터라 한다. 파일 포인터는 파일의 위치나 여러 정보가 저장되어 있는 곳을 가리키는 변수이다. 파일 포인터의 선언은 다음과 같이 한다.

● 선언 형식 : FILE * 파일 포인터 변수명

예) FILE * fp1, fp2 ;

2) 파일 열기와 닫기 : fopen() 과 fclose()

파일 포인터 변수가 생성되었으면 그 변수와 특정한 파일을 연결시켜 주어야 하는데, fopen()함수는 입출력할 파일을 열어 기준에 선언해 둔 파일 포인터 변수와 연결시킨다. 파일을 열 때는 파일 읽기를 할 것인가 혹은 쓰기를 할 것인가를 결정을 해야 한다. 파일 입출력 작업이 모두 끝나면 열어 둔 파일을 fclose()함수를 사용하여 닫아야 한다. 다음은 fopen()함수와 fclose() 함수의 형식이다.

● `fopen()` 함수와 `fclose()` 함수의 형식

```
FILE *파일포인터명 ;
```

```
파일포인터명 = fopen(파일명, 파일모드) ;
```

```
:
```

```
fclose(파일포인터명) ;
```

- 파일 명 : 열고자 하는 파일

- 파일 모드 : 입력과 출력 방식을 설정하는 모드임 (용도 설정)

파일은 반드시 용도에 맞게 파일 모드를 정하여 사용해야 한다.

〈표 I .4.2〉 파일 모드

종 류	의 미
"r"	파일을 처음 위치부터 읽기 용도로만 사용하기 위해 오픈하는 것으로, 반드시 파일이 사전에 생성되어 있어야 된다.
"w"	파일을 처음 위치부터 쓰기 용도로만 사용하기 위해 오픈하는 것으로, 파일이 존재하면 내용을 모든 내용을 지우고 기록을 하며, 파일이 없으면 새로 생성한다.
"a"	파일의 끝 부분에 추가하기 위하여 오픈하는 것으로 파일이 존재하지 않으면 새로 생성한다.
"r+"	파일을 읽고 쓰기 모두 가능하게 오픈하는 것으로, 파일은 반드시 존재해야만 한다.
"w+"	파일을 "w"의 기능에 읽기 기능을 추가한 것으로 읽기, 쓰기가 모두 가능하게 오픈하며 파일의 내용이 있으면 모두 지우고 처음부터 새로 기록을 하고, 파일이 존재하지 않으면 생성하고 기록한다.
"a+"	파일의 끝 부분에서 읽기와 쓰기를 하기 위한 용도로 오픈하며, 파일이 존재하지 않으면 새롭게 생성하고, 파일이 존재하면 파일의 끝에 추가한다.

예를 들어 다음과 같이 선언이 되었다면,

```
FILE *fp1 // 파일 포인터 변수 fp1 선언
```

```
fp1=fopen("input01.txt","r") ; // 파일을 읽기 용도로 오픈
```

```
:
```

```
fclose(fp1) ; // 파일을 닫음
```

"input01.txt"라는 파일은 읽기 전용으로 열리게 된다. 만약, 파일이 존재하지 않으면 `fopen()`함수는 NULL값을 되돌려 준다. 마지막의 `fclose()`함수는 파일을 닫아주는 역할을 한다. 쓰기 모드일 경우에는 파일의 끝에 파일의

끌을 알려주는 기호(EOF)를 불이고 닫는다. 파일이 정상적으로 닫혔을 경우엔 0을, 그렇지 않으면 1을 되돌려 준다.

3) getc()함수와 putc()함수

getc()함수는 getchar()함수와 putc()는 putchar()함수와 거의 같은 역할을 하는 것으로, 표준 입출력이 아닌 파일에 입출력을 한다는 점이 다르다.

getc()함수는 한 문자를 파일에서 읽어오는 것이고, putc()함수는 한 문자를 파일에 출력하는 함수이다. 참고로, fgetc()함수와 fputc()함수는 사용법과 결과가 같은 똑같은 함수라 할 수 있다. 형식은 다음과 같다.

● 형식 : **getc(파일 포인터 명);**
fgetc(파일 포인터 명);
putc(변수, 파일 포인터 명);
fputc(변수, 파일 포인터 명);

다음은 입력 파일에서 한 문자를 읽어 출력 파일에 기록하는 프로그램의 일부분이다.

```
char ch;
FILE *fp1, *fp2; // 파일 포인터 변수 선언
fp1= fopen("input1.txt", "r"); // 파일 열기
fp2=fopen("output1.txt", "w");
ch=getc(fp1); // fp1이 지정하는 파일에서 한 문자를 읽어 변수ch에 저장함
putc(fp2, ch); // 변수ch에 저장된 문자를 fp2가 지정하는 파일에 기록함
```

4) fgets()함수와 fputs()함수

fgets()함수는 지정된 파일로부터 다음 줄로 내리는 기호 '\n'을 만날 때까지의 문자열을 입력받는 함수이고, fputs()함수는 지정한 파일로 하나의 문자열을 기록하는 함수이다.

● 형식 : **fgets(변수, size, 파일 포인터명);**
fputs(변수, 파일포인터명);

여기서 변수는 일반적으로 문자열의 시작을 가리키는 포인터나 배열로 지정할 경우엔 배열명이 입력되는 것으로 저장 공간(버퍼)을 말한다. size는 저장 공간의 크기를 말한다.

5) fscanf()함수와 fprintf()함수 : 서식 있는 입출력 함수

fscanf() 함수는 파일 포인터가 지정하는 파일에서 서식에 맞게 읽어 들이는 함수이고, fprintf()함수는 서식 있는 문자열을 파일 포인터가 지정하는 파일에 출력하는 함수이다. 가장 많이 사용하는 입출력 함수이다.

일반적인 형식은 다음과 같다.

● 형식 : fscanf(파일포인터명, format, 인자의 주소);

fprintf(파일 포인터명, format, 인자);

format에는 입출력 양식이 들어가며, 당연히 fscanf()함수에서 입력 변수의 주소를 나타내는 포인터가, fprintf()문에서는 출력할 인자들이 들어간다.



문제 I .4.1

파일 “input01.txt”에서 4개의 점수를 읽어 평균을 “output01.txt”에 출력하는 프로그램을 작성해 보자.(단, 출력은 소수점 셋째자리에서 반올림하여 소수점 둘째자리까지 표현할 것)

입력 예) 80 90 100 94

출력 예) Average = 91.00

<풀이>

```
#include <stdio.h>
void main()
{
    FILE *fp1, *fp2 ;
    int a, b, c, d ;
    fp1 = fopen("input01.txt", "r") ;
    fp2 = fopen("output01.txt", "w") ;
    fscanf(fp1, "%d %d %d %d", &a, &b, &c, &d) ;
    fprintf(fp2, "Average = %5.2f\n", (float)(a+b+c+d)/4 ) ; // (float)는 형 변환
    fclose(fp1); fclose(fp2) ; // 파일 닫기는 생략 가능
}
```

파일 입출력은 이 정도로 마치고 앞으로 제어문이나 함수 단원에서 파일 입출력 문장에 대해 계속 연습을 할 것이다. 이 단원의 연습문제도 생략한다.

5. 디버깅

프로그램을 하다 보면 프로그램의 고장(failure)을 일으키는 에러(error) 또는 버그(bug)들을 찾게 되는데, 이런 에러를 찾아 올바르게 고치는 작업을 디버깅(debugging)이라고 한다. 에러는 문법적인 에러와 논리적인 에러가 있다. 문법적인 에러는 컴파일을 하면 Visual C++이 대부분 알려주거나, 도움말을 통해 쉽게 고칠 수 있다. 하지만, 논리적인 버그는 쉽게 찾을 수 없다. Visual C++에서는 디버깅 기능을 제공하여 에러를 찾는데 많은 도움을 주고 있다.

디버깅 방법에 대해 살펴보자.

가. 디버깅 모드 실행하기

디버깅은 중단점을 설정한 뒤 디버깅 모드를 실행시켜 에러를 찾고 마지막으로 디버깅 모드를 중단시켜야 한다. 디버깅 하는 절차는 다음과 같다.

- ① 프로그램에서 자신이 멈추고자 하는 위치에 커서를 두고 F9키를 눌러 중단점(Breakpoint)을 설정(또는 해제)한다. 왼쪽 여백에 갈색 점이 나타난다.



[그림 I .5.1] 빌더 도구막대

- ② F5키를 눌러 디버깅 모드를 실행시키면 메뉴 표시줄에 [Debug] 메뉴가 생성되며 중단점에서 실행(노란색 화살표)이 멈춘다. 또한 디버그 도구 상자도 나타난다.



[그림 I .5.2] 디버그 메뉴



[그림 I .5.3] 디버그 도구

- ③ 변수들의 실행 결과는 화면 아래쪽의 Variables(변수) 창과 Watch(와치) 창에서 확인할 수 있다. **Variables** 창에는 현재 실행 중인 라인의 변수들의 값만 나타낸다. **Watch** 창에는 특정한 변수에 대해 값이 변화되는 과정을 보고자 할 때 Name란에 더블클릭하여 변수명을 입력을 해 두면 된다. 변수관리 목록이라고 할 수 있다. 또한 Variables(변수) 창과 Watch(와치) 창에서 변수의 내용을 임의적으로 변경하여 특정한 상황을 만드는 것도 허용한다.(다양한 확인창은 [View]-[Debug Windows] 메뉴 이용 또는 디버그 도구 모음의 2번째 줄의 아이콘을 누르면 된다.)

Context: main()	
Name	Value
a	10
k	10
sub1 returned	10
Auto	
Locals	
this	

Name	Value
a	10
k	10

[그림 I .5.4] Variables창과 Watch창

그 밖에 콜 스택(Call Stack) 창은 함수가 호출되기까지의 모든 호출원을 보여 주고, 메모리(Memory) 창은 큰 배열이나 문자열, 이진 데이터처럼 일정한 형태가 없는 값 등을 확인할 수 있도록 해 준다.

- ④ 다음의 메뉴들 중 한 가지를 이용하여 한 줄씩 또는 해당 위치로 옮겨 실행하면 된다. (디버그 메뉴 또는 디버그 도구, 단축키 이용)
- [Step Into] : 함수를 호출하는 경우 함수 내부로 들어가 한 줄씩 실행 (단축키 : F11)
 - [Step Over] : 함수를 호출하는 경우, 함수 내부로 들어가지 않고 한 줄씩 실행하며 보통 가장 많이 사용 (단축키 : F10)
 - [Step Out] : 원하지 않은 함수로 들어온 경우, 함수 밖으로 빠져 나옴 (단축키 : Shift + F11)
 - [Run to Cursor] : 현재 지정한 커서의 위치까지 실행 (단축키 : Ctrl + F10)
 - Set Next Statement : 마우스 오른쪽의 팝업메뉴에 있으며, 사용자가 원하는 위치로 실행 지점을 변경

- ⑤ 만약, 기본적으로 제공하는 라이브러리 함수 내부로 들어가면 Find Source 창이 나타난다. 이런 경우는 “Cancel”을 클릭한 뒤 단축키 Shift + F11 키를 눌러 빠져나오면 된다.
- ⑥ [Debug]-[Quickwatch] 메뉴를 실행시켜 빠르게 원하는 변수 값을 확인 할 수 있다. Quickwatch 대화 상자가 나타나면 “Expression”란에 찾고자 하는 변수명을 입력하고 “Recalculate” 버튼을 누르면 변수값이 나타난다. “Watch” 버튼을 누르면 Watch 창에 변수가 등록된다.
- ⑦ 디버깅을 마치려면 [Debug]-[Stop Debugging] 또는 단축키 Shift + F5를 이용하면 된다.



문제 I .5.1

중단점 설정 및 해제를 할 때 사용하면 편리한 단축기는 무엇인가?

- ① F5 ② Shift+F5 ③ F9 ④ Shift+F9 ⑤ F10

<풀이> ③



문제 I .5.2

특정한 변수를 계속해서 관리하려면 어떤 창을 띄워두는 것이 좋은가?

- ① Watch창 ② Call창 ③ Memory창 ④ Variables창 ⑤ output창

<풀이> ①

6. 제어문

프로그램의 수행은 특별한 제어가 없다면, 일반적으로 처음부터 끝까지 순서대로 수행이 된다. 경우에 따라서는 흐름을 다른 곳으로 이동시켜 주거나 뒤로 되돌려 주는 등 다양한 방법으로 순서를 제어해야 하는데, 이런 역할을 하는 명령문을 제어문이라고 한다. 제어문은 크게 다음 세 종류로 나눌 수 있다.

- 선택문(조건문) : if문, if~else문, switch~case문
- 반복문(순환문) : for문, while문, do~while문
- 분기문(점프문) : break문, continue문, return문

가. 선택문

특정한 조건의 결과에 따라 하나의 선택을 하도록 하는 명령어를 선택문이라고 한다. if문, if~else문과 switch~case문 등이 있다.

1) if ~ else문

주어진 조건이 참이면 if문에 속한 문장들만 수행하고, 조건이 거짓이면 else문에 속한 문장들만 수행하는 명령어이다. 복합 if문 안에 또 다른 if문을 하나 이상 포함시키는 것을 복합 if문이라고 하며, 여러 개의 조건을 고려하기 때문에 선택의 방향이 훨씬 더 많아진다.

〈표 I .6.1〉 if문의 기본 구조

if~else문 기본 구조	복합 if문 기본 구조
<pre>if(조건) 문장1 ; else 문장2 ; 문장3 ;</pre>	<pre>if(조건1) 문장1 ; else if(조건2) 문장2 ; else 문장3 ; 문장4 ;</pre>

다음은 숫자 N를 입력받아 N이 홀수인지 짝수인지를 구분하는 프로그램이다.

```
#include <stdio.h>
void main( )
{
    int data ;
    scanf("%d", &data) ;
    if( (data%2) == 0 ) printf("짝수 \n") ;
    else printf("홀수 \n") ;
}
```



문제 I .6.1

다음은 숫자를 입력받아 3의 배수인지 2의 배수인지, 그 외의 숫자인지를 나타내는 프로그램을 작성하시오.

```
#include <stdio.h>
void main( )
{
    int data ;
    scanf("%d", &data) ;
    if(data%3 == 0 && data%2 == 0 ) printf("3의 배수이면서 2의 배수 \n") ;
    else if(data%3 == 0 && data%2 != 0 ) printf("3의 배수 \n") ;
    else if(data%3 != 0 && data%2 == 0) printf("2의 배수 \n") ;
    else printf(" 그 외의 숫자 \n") ;
}
```

2) switch ~ case문

하나의 조건에 대해 경우의 수가 3개 이상일 때 사용하는 명령문이다. 즉 조건에 따라 해당 경우를 찾아 관련된 문장들을 수행하는 것이다.

〈표 I .6.2〉 switch~case문의 기본 구조

기본 구조
<pre>switch(조건) { case 경우1 : 문장1 ; break ; case 경우2 : 문장2 ; break ; case 경우3 : 문장3 ; break ; default : 문장4 ; } 문장5 ;</pre>

switch~case문에서 break문은 정해진 범위를 빠져나가는 것을 의미하는 문장으로서 만약 break문을 생략하면 해당 경우로 들어가서 문장들을 수행하고, 아래쪽에 있는 모든 경우의 문장들을 수행하게 된다. default는 조건에 맞는 경우가 없을 때 수행된다.



문제 I .6.2

두 숫자와 사칙연산 기호를 입력받아 계산하는 프로그램을 작성하시오.

입력 예) 50 * 2

출력 예) 50 * 2 = 100

<풀이>

```
#include <stdio.h>
void main( )
{
    int data1, data2 ;
    char op ;
    scanf("%d %c %d", &data1, &op, &data2) ;
    switch(op) {
        case '+':
            printf(" %d + %d = %d", data1, data2, data1+data2) ;
            break;
        case '-':
            printf(" %d - %d = %d", data1, data2, data1-data2) ;
            break;
        case '*':
            printf(" %d * %d = %d", data1, data2, data1*data2) ;
            break;
        default :
            printf(" %d / %d = %f", data1, data2, (float)(data1/data2) ) ;
    }
}
```

나. 반복문

프로그램의 일부분을 반복하여 수행하도록 하는 것을 반복문 또는 순환문이라고 한다. 반복문의 종류에는 for문, while문, do~while문 등이 있다. 반복문은 경우에 따라 if문과 break문, continue문 등의 분기문을 이용하여 제어를 하는 경우도 많다.

1) for문

반복문에서 가장 많이 사용되는 제어문으로, 초기값이나 최종값 또는 반복 횟수가 지정된 경우에 많이 사용된다. 수행 순서는 처음에 초기값을 정하고, 조건식을 수행한 다음 조건이 참이면, 내부 문장들을 수행한다. 이제 증감값을 정하고 다시 조건식을 수행하여 참이면 계속 반복하고, 거짓이 나오면 for문을 빠져 나가는 것이다. 반복 수행할 문장이 하나이면 {}는 생략할 수 있다. 형식은 <표 I .6.3>과 같다.

<표 I .6.3> for문의 기본 구조

기본 구조	순 서 도
<pre>for(초기값 ; 조건식 ; 증감값) { 문장들 ; }</pre> <ul style="list-style-type: none"> □ 초기값 : 반복을 제어하는 제어변수의 초기값을 지정 (처음에 한번만 수행함) □ 조건식 : 반복문을 수행하기 위한 조건문, 조건이 참이면 문장들을 수행하고, 거짓이면 for문을 빠져나감 □ 증감값 : 제어변수의 증가 혹은 감소의 변화량 □ 초기값, 조건식, 증감값은 생략할 수도 있음 	<pre> graph TD A[초기값] --> B{조건식} B -- 예 --> C[문장들] C --> D[증감값] D --> B B -- 아니오 --> E[문장들] </pre>

예제로 n부터 m까지의 합을 구하는 프로그램을 살펴보자.(단, m>n임)

```
#include <stdio.h>
void main( )
{
    int i, n, m, sum=0 ;
    scanf("%d %d", &n, &m) ;
    for(i=n; i<=m; i++) sum=sum+i ;
    printf(" %d부터 %d까지 합 : %d\n", n, m, sum) ;
}
```



문제 I .6.3

다음은 어떤 프로그램의 일부이다. 프로그램을 실행시킨 후 i의 값은?

```
int i, c=0 ;
for(i=1 ; i<=30 ; i+=7) c++ ;
```

- ① 22 ② 29 ③ 30 ④ 36 ⑤ 40

<풀이> ④



문제 I .6.4

10부터 N까지 3의 배수와 4의 배수가 아닌 정수들의 합을 구하는 프로그램을 작성하시오.(단,N>10)

<풀이>

```
#include <stdio.h>
void main( )
{
    int i, N, sum=0 ;
    scanf("%d", &N) ;
    for(i=10; i<=N; i++) if( (i%3!=0) && (i%4!=0) ) sum+=i ;
    printf("%d \n", sum) ;
}
```

2) while문

while문의 조건이 참이면 안쪽의 문장들을 수행하고 거짓이 되면 while문을 빠져나간다. 문장이 하나면 {}를 생략할 수 있다. 형식은 다음과 같다.

〈표 I .6.4〉 while문의 기본 구조

기본 구조	순 서 도
<pre>while(조건식) { 문장1 ; } 문장2 ;</pre>	<pre> graph TD A{조건식} -- 예 --> B[문장 1] A -- 아니오 --> C[문장 2] </pre>

만약, while문의 조건이 항상 참이면 무한 반복이 된다.

10부터 n까지 합을 구하는 프로그램을 while문으로 작성하면 다음과 같다.

```
#include <stdio.h>
void main( )
{
    int i=10, n, sum=0 ;
    scanf("%d", &n) ;
    while(i<=n) {
        sum+= i ;
        i++ ;
    }
    printf(" 10부터 %d까지의 합 : %d\n", n, sum) ;
}
```



문제 I 6.5

input01.txt파일에서 입력자료 개수와 입력자료를 읽어와 최대값과 최소값의 차이를 output01.txt 파일로 출력하시오.(단, while문으로 작성할 것)

입력 예)

5

20 30 25 19 40

출력) 21

<풀이>

```
#include <stdio.h>
void main( )
{
    FILE *fp1, *fp2 ;
    int i=0, n, data, max, min ;
    fp1=fopen("input01.txt", "r") ;
    fp2=fopen("output01.txt", "w") ;
    fscanf(fp1, "%d", &n) ;
    while(i<n) {
        fscanf(fp1, "%d", &data) ;
        if(i==0) max=min=data ;
        if(max<data) max=data ;
        else if(min>data) min=data ;
        i++ ;
    }
    fprintf(fp2, "%d\n", max-min) ;
    fclose(fp1); fclose(fp2) ;
}
```

3) do~while문

do~while문은 조건문을 마지막에 둘으로써 최소한 1번 이상 반복 수행할 수 있도록 하는 명령문이다. do문 안의 문장들을 한번 수행하고 조건식을 검사한 후 참이면 위로 올라가 반복해야 할 문장을 수행한다. 조건식이 거짓이 나올 때까지 계속 반복한다. while()뒤에 반드시 세미콜론(;)이 붙어야 한다.

〈표 I .6.5〉 do~while문의 기본 구조

기본 구조	순서도
<pre>do { 문장1 ; } while(조건식) ; 문장2 ;</pre>	<pre> graph TD A[문장 1] --> B{조건식} B -- 예 --> C[문장 2] B -- 아니요 --> A </pre>

예로 10부터 n까지의 합을 구하는 프로그램을 do~wile문을 이용해서 작성해 보자.(단, n>10)

```
#include <stdio.h>
void main( )
{
    int i=10, n, sum=0 ;
    scanf("%d", &n) ;
    do {
        sum+=i ;
        i++ ;
    } while(i<=n) ;
    printf(" 10부터 %d 까지의 합계 : %d \n", n, sum) ;
}
```

4) 분기문

현재의 실행 부분에서 문장의 다른 특정 부분으로 실행을 옮기려고 할 때 사용하는 것을 분기문이라 한다. break문, continue문, goto문 등이 있다.



연습문제

1

다음 중 구조적인 프로그램(C언어) 개발에서 조건에 상관없이 한 번 이상 무조건 실행하는 문장은?(2005 교원)

- ① for문
- ② while문
- ③ switch문
- ④ if문
- ⑤ do-while문

2

1부터 10까지 홀수의 합을 구하는 프로그램이 아닌 것은?(2005 중등)

① s = 0 ;

```
for ( i = 1; i <= 10; i += 2 )
    s = s + i ;
```

② s = 0 ;

```
for ( i = 1; i <= 5; i++ )
    s = s + i * 2 - 1 ;
```

③ s = 0 ;

```
for ( i = 1; i <= 10; i++ )
{
    s = s + i ;
    i = i + 1 ;
}
```

④ i = -1; s = 0 ;

```
while ( i < 10 )
{
    i = i + 2 ;
    s = s + i ;
}
```

⑤ i = 0 ; j = -1 ; s = 0 ;

```
while ( i < 5 )
{
    i = i + 1 ;
    j = j + 2 ;
    s = s + j ;
}
```

3

10부터 1000까지 N으로 나눈 나머지를 중 홀수들만 합을 구하시오.

4

두 개의 양수를 입력받아 두 숫자 중 큰 수를 계속 합하려고 한다. 단, 두 숫자 중 하나라도 0이 입력되면 계산을 종료하고 결과를 출력하시오.



가로 N행, 세로 N열에 대해 아래와 같이 '#'표시가 역삼각형의 형태로 출력되도록 만드시오.

입력) N=3 일 때

출력 예)

```
# # #
# #
#
```



10이상 40미만의 나이를 입력받아 10대, 20대, 30대 인지를 나타내는 프로그램을 작성하시오.

입력) 25

출력) 25는 20대이다.



다음은 무엇을 구하는 프로그램인가?(2005 중등)

```
c = N - 1 ;
for ( i = 2; i <= N; i++ )
    for ( j = 2; j < i; j++ )
        if ( i % j == 0 )
        {
            c-- ;
            break ;
        }
printf( "%d \n", c ) ;
```

- | | |
|----------------|---------------|
| ① N의 약수의 개수 | ② N이하인 소수의 개수 |
| ③ N이하인 홀수의 개수 | ④ N이하인 짝수의 개수 |
| ⑤ N이하인 합성수의 개수 | |



<그림 1>과 같이 9×9 격자판에 쓰여진 81개의 자연수가 주어질 때, 이들 중 최대값을 찾고 그 최대값이 몇 행 몇 열에 위치한 수인지 구하는 프로그램을 작성하시오. (2007 고등)

예를 들어, 다음과 같이 81개의 수가 주어지면

	1열	2열	3열	4열	5열	6열	7열	8열	9열
1행	3	23	85	34	17	74	25	52	65
2행	10	7	39	42	88	52	14	72	63
3행	87	42	18	78	53	45	18	84	53
4행	34	28	64	85	12	16	75	36	55
5행	21	77	45	35	28	75	90	76	1
6행	25	87	65	15	28	11	37	28	74
7행	65	27	75	41	7	89	78	64	39
8행	47	47	70	45	23	65	3	41	44
9행	87	13	82	38	31	12	29	29	80

〈그림 1〉

이들 중 최대값은 90이고, 이 값은 5행 7열에 위치한다. 실행파일의 이름은 MAX.EXE로 하고, 프로그램의 실행시간은 1초를 넘을 수 없다. 부분 점수가 주어질 수 있다.

〈입력 형식〉 입력 파일의 이름은 INPUT.TXT로 한다. 첫 째 줄부터 아홉 번째 줄까지 한 줄에 아홉 개씩 자연수가 주어진다. 주어지는 자연수는 100보다 작다.

〈출력 형식〉 출력 파일의 이름은 OUTPUT.TXT로 한다. 첫째 줄에 최대 값을 출력하고, 둘째 줄에 최대값이 위치한 행 번호와 열 번호를 빈칸을 사이에 두고 차례로 출력한다. 최대값이 두 개 이상인 경우 그 중 한 곳의 위치를 출력한다.

〈입력과 출력의 예〉

입력파일(INPUT.TXT)

```
3 23 85 34 17 74 25 52 65
10 7 39 42 88 52 14 72 63
87 42 18 78 53 45 18 84 53
34 28 64 85 12 16 75 36 55
21 77 45 35 28 75 90 76 1
25 87 65 15 28 11 37 28 74
65 27 75 41 7 89 78 64 39
47 47 70 45 23 65 3 41 44
87 13 82 38 31 12 29 29 80
```

출력파일(OUTPUT.TXT)

```
90
5 7
```



풀



① ⑤

② ④

③

```
#include<stdio.h>
void main()
{
    int i, n, sum=0, temp ;
    scanf("%d", &n) ;
    for(i=10; i<=1000; i++){
        temp=i%n ;
        if(temp%2 != 0) sum+=temp ;
    }
    printf("result : %d \n", sum) ;
}
```

④

```
#include<stdio.h>
void main()
{
    int n, m, sum=0 ;
    while(1){
        scanf("%d %d", &n, &m) ;
        if(n==0 || m==0) break ;
        sum += ( (n>m)? n : m ) ;
    }
    printf("sum : %d \n", sum);
}
```



```
#include<stdio.h>
void main()
{
    int i, j, n ;
    scanf("%d", &n) ;
    for(i=n; i>0; i--){
        for(j=n-1; j>=i; j--) printf(" ") ;
        for(j=i; j>0; j--) printf("# ") ;
        printf("\n") ;
    }
}
```



```
#include<stdio.h>
void main()
{
    int d ;
    scanf("%d", &d) ;
    switch(d/10) {
        case 3 : printf(" %d는 %d대 \n", d , d/10*10) ;
        break ;
        case 2 : printf(" %d는 %d대 \n", d , d/10*10) ;
        break ;
        case 1 : printf(" %d는 %d대 \n", d , d/10*10) ;
        break ;
        default : printf(" 10대 ~ 30대가 아닙니다. \n") ;
    }
}
```



②



```
#include <stdio.h>
void main()
{
    int row=0, col=0, max=0 ;
    int i, j, a ;
    FILE *fp1, *fp2 ;
    fp1 = fopen("input.txt", "r") ;
    for(i=1; i<=9; i++) for (j=1; j<=9; j++) {
        fscanf(fp1, "%d", &a) ;
        if (a>max) {
            max=a ;
            col=j ;
            row=i ;
        }
    }
    fp2 = fopen("output.txt", "w") ;
    fprintf(fp2, "%d \n %d %d \n", max, row, col) ;
    fclose(fp1) ;  fclose(fp2) ;
}
```

7. 함수

프로그래밍을 보다 쉽고 편리하게 하기 위해서는 프로그램을 논리적으로 작은 단위로 나누어 프로그래밍을 하게 된다. 이런 프로그램의 논리적 작은 단위를 **함수(function)**라고 한다. 함수는 크게 시스템 정의 함수와 사용자 정의 함수로 나눌 수 있는데, 시스템 정의 함수는 라이브러리라고도 하며 printf() 등과 같이 시스템에서 이미 정의가 되어 있어서 바로 호출하여 사용하면 되고, 사용자 정의 함수는 사용자의 필요에 따라 직접 정의하여 사용한다.

가. 함수의 기본적인 표현 구조

함수 정의(definition) 부분이 호출문보다 앞에 있는 구조와 함수 **프로토타입(prototype)**을 호출문보다 앞에 선언하고 함수 정의 부분은 호출문 뒤에 있는 구조로 나눌 수 있다. 함수 프로토타입은 전처리기에 의해 컴파일하기 전에 먼저 처리를 하도록 되어 있다. 함수의 구성은 다음 <표 I .7.1>와 같다.

<표 I .7.1> 함수의 기본 표현

함수의 정의 구현 부분 먼저	함수 프로토타입 이용
<pre>자료형 함수이름(가인수) { 문장들 ; return (되돌림 값) ; } void main() { 문장들 ; 함수 호출(실인수) ; }</pre>	<pre>자료형 함수이름(가인수); void main() { 문장들 ; 함수 호출(실인수) ; 자료형 함수이름(가인수) { 문장들 ; return (되돌림 값) ; } }</pre>
<ul style="list-style-type: none"> □ 함수를 호출하는 문장보다 함수의 정의가 먼저 이루어져야 함. 	<ul style="list-style-type: none"> □ 함수 프로토타입을 호출문보다 먼저 선언하여 정의 부분이 있다는 것을 표시함.

- 함수 정의 부분의 함수 이름 앞의 자료형은 return문에 의해 호출문으로 되돌려지는 결과값의 자료형을 나타낸다.
- 호출문의 인자를 함수 호출할 때 넘어가는 실인수(real parameter)라 하고, 호출되는 함수의 인자는 실인수의 값을 받는 가인수(formal parameter)라고 한다. 결과적으로 실인수의 값이 가인수에 전달되는 것이다.
- return문의 반환값은 결과값, 간단한 수식, 다른 함수 등도 표시할 수 있다.

함수 호출문은 다음 <표 I .7.2>과 같은 형식으로 사용한다.

<표 I .7.2> 함수 호출문 형식

형식	설명	사용 예
함수명(인자1, 인자2, ……, 인자n) ;	반환되는 값이 없음	sum(n) ;
변수 = 함수명(인자1, 인자2, …, 인자n) ;	반환되는 값을 저장	result = sum(n) ;

함수 프로토타입의 표현은 다음과 같이 인자의 자료형과 인자명을 모두 표시하는 경우와 인자의 자료형만 표시하는 경우로 나눌 수 있다.

● 형식 :

함수반환값의 자료형 함수이름(자료형 인자1, …, 자료형 인자n) ;

또는 함수반환값의 자료형 함수이름 (자료형, 자료형, …, 자료형) ;

나. 인자의 전달 방식

여러 가지 방식이 있지만 많이 사용하는 값에 의한 호출(call by value)와 참조에 의한 호출(call by reference)만 보도록 하자.

1) 값에 의한 호출

호출 함수의 인자인 실인수의 실제 값을 호출되는 함수의 가인수에 전달하는 방식이다. 실인수와 가인수가 같은지만, 다른 공간 배치된 별개의 변수이다. 따라서, 가인수의 값이 바뀌어도 실인수의 값에는 변화가 없다.

다음 예를 통해 알아보자.

```
#include <stdio.h>
float average(int , int) ; // 함수 프로토타입
void main()
{
    int a, b ;
    float av ;
    scanf("%d %d", &a, &b) ;
    av = average(a, b) ; // 함수 호출문 - 실인수는 값을 전달
    printf(" 두 수의 평균 : %10.2f", av) ;
}
float average(int x, int y) // 함수 정의부 - 가인수는 값을 받음
{
    return (float)(x+y)/2 ; // return문에 의해 결과를 되돌려 줌
}
```

호출문의 실인수 a와 b는 결과값을 호출되는 average()함수 정의 부분의 x, y에 값을 전달되고 return문에 의해 결과를 되돌려진다.

2) 참조에 의한 호출

참조에 의한 호출은 값에 의한 호출처럼 값을 전달하는 것이 아니라, 실인수의 주소를 전달함으로써 가인수는 주소를 받아 실인수와 같은 공간을 사용하도록 되는 것이다. 만약, 함수에서 가인수의 값이 바뀌면 실인수의 값도 내용이 바뀌게 된다. 다음 예제를 통해 살펴보자.

```
#include <stdio.h>
void average(int *, int *) ; //함수 프로토타입
void main()
{
    int a, b ;
    scanf("%d %d", &a, &b) ;
    average(&a, &b) ; //함수 호출부 - 실인수는 주소를 넘겨줌
}
void average(int *x, int *y) //함수 정의부 - 가인수는 주소를 받음
{
    float av ;
    av = (float) (*x + *y) / 2 ; // *x는 x가 지시하는 변수 a의 값
    printf(" 두 수의 평균 : %10.2f", av) ; // return문 없음
}
```

호출문에서 변수 a와 b의 주소를 넘겨주고 호출되는 average()함수에서는 가인수가 주소를 받을 수 있는 포인터 변수로 선언이 되어야 한다. x는 a를 가리키고, y는 b를 가리키고 있어서 가인수 x와 y의 값이 바뀌면 실인수의 a와 b의 내용이 바뀐다. *x + *y 는 x가 가리키는 값(x의 값)과 y가 가리키는 값(y의 값)을 더하라는 말이다.

다. 변수의 범위 (지역변수와 전역 변수)

변수는 사용할 수 있는 유효 범위에 따라 전역 변수와 지역변수로 나눈다.

1) 지역변수

지역변수는 함수의 내부에서 선언되어 함수 내에서만 사용이 되는 변수이며, 내부 변수라고도 한다. 변수는 변수가 선언될 때 메모리 공간이 부여되고, 함수가 끝나면 메모리 공간이 없어진다.

2) 전역변수

전역변수는 함수 외부에 선언되어 선언된 이후의 모든 함수에서 사용이 가능한 변수를 말하며, 외부변수라고도 한다. 프로그램이 실행이 시작될 때 선언이 되어 끝날 때까지 계속 메모리 공간이 남아 있다. 만약, 지역변수와 전역변수의 변수명이 같을 경우는 지역변수가 우선이다.

다음 예를 살펴보고, 지역 변수와 전역변수의 결과값을 살펴보자.

```
#include<stdio.h>
int k=50 ; // 전역변수 k 선언
void abc()
{
    int a=5 ; //지역변수 a 선언
    printf("abc()함수 : a=%d, k=%d\n", a, k) ;
    //지역변수 a, 전역변수 k값 출력
}
```

```

void main()
{
    int a=10, k=20 ;                      // 지역변수 a, k 선언
    k=k+a ;
    printf("main()함수 : a=%d, k=%d\n", a, k) ; // 지역변수 a와 k값 출력
    abc() ;
}

```

실행 결과

실행 결과	main()함수 : a=10, k=30 abc()함수 : a=5, k=50
--------------	--

라. 되부름 함수

자기 자신을 부르는 함수를 **되부름 함수(recursive function)** 또는 재귀적 호출 함수라고 하고, 보통 동일한 기능의 반복을 처리하기 위해 사용한다. 1 1 2 3 5 8 과 같은 피보나치 수열은 보통 되부름 함수로 표현을 많이 한다. 다음 예의 결과는 얼마인지 계산해 보자.(2002 경남)

```

#include<stdio.h>
int f(int n, int k)
{
    if(k==1 || n==k) return 1 ;
    else return k*f(n-1, k) + f(n-1, k-1) ;
}
void main()
{
    printf("%d\n", f(5, 3) ) ;
}

```

f(5,3)을 호출하면 f()함수의 return문에 의해 $3*f(4,3)+f(4,2)$ 를 되돌려준다. 다시 f(4,3)과 f(4,2)는 각각의 f()함수를 호출하게 된다. 결국 f(1,0)이 되면 1을 되돌려준다. 따라서 출력은 25가 된다.



문제 I .7.1

다음 함수에서 recursive(5)를 호출할 때 출력되는 "*"의 개수는? (2006 교원)

```
void recursive(int n)
{
    if(n>1) {
        recursive(n/2) ;
        recursive(n/2 );
    }
    printf("*");
}
```

- ① 4 ② 5 ③ 6 ④ 7 ⑤ 8

<풀이> ④



문제 I .7.2

다음 프로그램의 실행 결과는 무엇인가? (2007 교원)

```
#include<stdio.h>
bool isFunction(int n) ;
void main()
{
    int i, j=0 ;
    for(i=0; i<6; i++) {
        if( isFunction(4*i+3) ) j = j + (4*i+3) ;
    }
    printf("%d", j) ;
}
bool isFunction(int n)
{
    for(int i=2; i<n; i++) {
        if(n%i==0) return false ;
    }
    return true ;
}
```

- ① 63 ② 40 ③ 78 ④ 55 ⑤ 50

<풀이> ①



연습문제

1

다음 프로그램을 실행시키면 어떠한 결과가 출력되는가?(2002 초등)

```
#include <stdio.h>
int f(int n)
{
    if (n <= 2) return n;
    else return f(n-1) * f(n-2);
}
void main()
{
    printf("%d\n", f(7));
}
```

- ① 64 ② 128 ③ 196 ④ 256 ⑤ 384

2

다음 함수는 무엇을 하는 함수인가? (2004 고등)

```
int f(int a, int b)
{
    return ((a + b) + abs(a - b)) / 2; // abs()함수는 절대값을 반환하는 함수
}
```

- ① 두 값 중 큰 값을 구하는 함수 ② 두 값 중 작은 값을 구하는 함수
 ③ 두 값의 평균값을 구하는 함수 ④ 두 값의 차를 구하는 함수
 ⑤ 한 값을 다른 한 값으로 나눈 나머지를 구하는 함수

3

다음 프로그램을 실행시켰을 때의 출력 결과는?(2007 교원)

```
#include <stdio.h>
int gcd(int m, int n);
void main(void)
{
```

```

int a=30, b=40, c;
if(a > b) c = gcd(a,b);
else c = gcd(b,a);
printf("%d",c);
}

int gcd(int m, int n)
{
    if(n==0) return m;
    else return gcd(n,m%n);
}

```

- ① 10 ② 15 ③ 20 ④ 25 ⑤ 30



한계값 N까지 친화수(완전수)의 쌍을 표현하고 쌍의 개수도 알려주는 프로그램을 작성하시오. 친화수가 서로 자신을 제외한 약수들의 합의 되는 수를 말한다. 예를 들면, 220은 약수의 합이 284이고 284의 약수의 합은 220으로 200과 284는 친화수이다. (단, N은 2000이하의 수)

입력(INPUT.TXT)

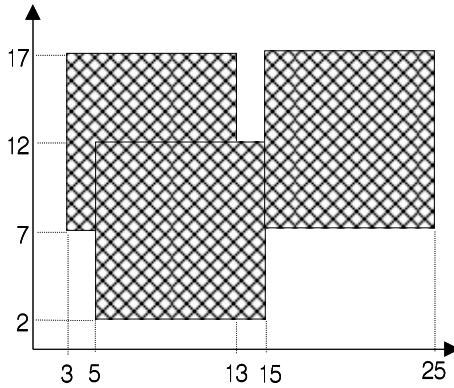
1500

출력(OUTPUT.TXT)

220, 284
284, 220
1184, 1210
1210, 1184
Total Count : 4



가로, 세로의 크기가 각각 100인 정사각형 모양의 흰색 도화지가 있다. 이 도화지 위에 가로, 세로의 크기가 각각 10인 정사각형 모양의 검은색 색종이를 색종이의 변과 도화지의 변이 평행하도록 붙인다. 이러한 방식으로 색종이를 한 장 또는 여러 장 붙인 후 색종이가 붙은 영역의 넓이를 구하는 프로그램을 작성하시오. (2007 초등)
예를 들어 흰색 도화지 위에 세 장의 검은색 색종이를 <그림 1>과 같은 모양으로 붙였다면 검은색 영역의 넓이는 260이 된다.



실행파일의 이름은 BB.EXE로 하고, 프로그램의 실행시간은 1초를 넘을 수 없다. 부분 점수는 없다.

<입력 형식> 첫째 줄에 색종이의 수가 주어진다. 이어 둘째 줄부터 한 줄에 하나씩 색종이를 붙인 위치가 주어진다. 색종이를 붙인 위치는 두 개의 자연수로 주어지는데 첫 번째 자연수는 색종이의 왼쪽 변과 도화지의 왼쪽 변 사이의 거리, 두 번째 자연수는 색종이의 아래쪽 변과 도화지의 아래쪽 변 사이의 거리이다. 색종이의 수는 100이하이며, 색종이가 도화지 밖으로 나가는 경우는 없다.

<출력 형식> 첫째 줄에 색종이가 붙은 겹은 영역의 넓이를 출력한다.

<입력과 출력의 예>

입력 (INPUT.TXT)

3
3 7
15 7
5 2

출력 (OUTPUT.TXT)

260



④

①

①

④

```
#include <stdio.h>
int fd(int) ;
void main()
{
    FILE *fp1, *fp2 ;
    int i, MAX, sum, count=0 ;
    fp1=fopen("input.txt", "r") ;
    fp2=fopen("output.txt", "w") ;
    fscanf(fp1, "%d", &MAX) ;
    for(i=2; i<=MAX; i++) {           //MAX 까지 검색
        sum = fd(i) ;
        if(i == fd(sum) && i!=sum && sum<=MAX) {
            // i의 약수합이 tempSum 약수의 합이 일치 and 같은수 제외
            fprintf(fp2, "%d , %d \n", i, sum) ;
            count++ ;
        }
    }
    fprintf(fp2, "Total Count : %d \n", count) ;
    fclose(fp1) ;
    fclose(fp2) ;
}
```

```
int fd(int d) //자신을 제외한 약수합 계산 함수
{
    int i ;
    int sum = 0 ;
    for(i=1; i<d; i++) if(d%i==0) sum+=i ;
    return sum ;
}
```



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

void main()
{
    int n, x, y, i, j ;
    int area=0 ;
    FILE *fp1, *fp2 ;
    bool p[101][101] ;
    memset(p, 0, sizeof p) ;           // 메모리의 내용을 0으로 초기화
    fp1 = fopen("input.txt", "r") ;
    fscanf(fp1, "%d", &n) ;
    while (n>0) {
        n-- ;
        fscanf(fp1, "%d%d", &x, &y) ;
        for (i=x; i<x+10; i++) for (j=y; j<y+10; j++) {
            p[i][j]=true ;
        }
    }
    fp2 = fopen("output.txt", "w") ;
    for (i=1; i<100; i++) for (j=1; j<100; j++) {
        if (p[i][j]) area++ ;
    }
    fprintf(fp2, "%d\n", area) ;
}
```

8. 배열

자료형이 같은 변수들을 하나로 묶어 관리하는 것을 **배열**이라고 하며, 메모리에 저장될 때는 연속해서 저장된다. 배열은 첨자의 개수에 따라 1차원, 2차원, 3차원 등으로 선언될 수 있다. 배열의 사용법에 대해 알아보자.

가. 1차원 배열

배열의 선언 형식은 다음과 같다.

● 배열 선언 형식 : **자료형 배열이름[크기]**

예를 들어 int jumsu[5] ; 형식으로 선언을 하면 메모리에 int형을 저장할 수 있는 공간 5개가 연속해서 마련된다. 결국, C 언어에서는 실제 변수의 첨자는 0부터 시작하여 jumsu[0]부터 jumsu[4]까지 5개 정수형 변수가 만들어진다.

배열 변수 선언 int jumsu[5] ;

jumsu[0]	jumsu[1]	jumsu[2]	jumsu[3]	jumsu[4]
----------	----------	----------	----------	----------

배열 변수도 형을 선언을 하면서 초기화를 할 수 있다.

```
int data[5] = {50, 60, 70, 80, 90} ;
char ch[6] = {'H', 'e', 'l', 'l', 'o', '\0'} ;
char ch[] = {'H', 'e', 'l', 'l', 'o', '\0'} ; //자동으로 크기를 6으로 정함
char ch[6]= "Hello" ; // 문자열의 마지막엔 '\0'(널)문자 포함됨
```

다른 예로, “Have a nice day!” 문장을 gets()함수를 이용해 키보드로 입력 받으려고 할 때 변수 선언은 아래와 같이 할 수 있다.

```
char ch[20] ;
```

```
gets(ch) ;
```

주어진 문장은 공백과 마지막 NULL 문자까지 포함하여 17개 문자를 저장할 수 있는 공간이 필요하다. 메모리에 저장된 형태를 보면 다음과 같다.

'H'	'a'	'v'	'e'		'a'		'n'	'i'	'c'	'e'		'd'	'a'	'y'	'!'	'\0'	
-----	-----	-----	-----	--	-----	--	-----	-----	-----	-----	--	-----	-----	-----	-----	------	--

출력은 `printf("%s",ch);` 또는 `puts(ch);` 형식으로 배열명만 입력하면 된다.
간단한 예로, 사람 이름을 입력받아 출력하는 프로그램을 작성해 보자.

```
#include <stdio.h>
void main( )
{
    char name[20] ;
    printf(" 이름을 입력하시오 : ") ;
    scanf("%s", name) ;
    printf(" 당신의 이름은 %s입니다. \n", name) ;
}
```

배열명은 배열의 시작을 가리키는 주소라고 생각하면 된다. 그래서 `scanf()`문에서 배열명 `name`만을 적어주고, `printf()`문에서는 `%s`와 대응되는 배열명 `name`만 적어주면 배열 `name`의 시작 주소부터 '`\0`'을 만날 때까지 문자들을 출력하게 된다.



문제 I .8.1

다음의 출력 결과는 무엇인가? (2006 교원)

```
int a[5]={7,9,5,8,9}, m1, m2, t, i ;
m1 = a[0] ;
m2 = a[0] ;
t = a[0] ;
for(i=1; i<=4; i++) {
    t += a[i] ;
    if(a[i] < m1) m1 = a[i] ;
    if(a[i] > m2) m2 = a[i] ;
}
t = (m1 + m2) ;
printf("%d", t) ;
```

- ① 20 ② 22 ③ 23 ④ 24 ⑤ 38

<풀이> ④

나. 2차원 배열

2차원 배열은 첨자의 개수가 2개이고, 행과 열을 나타낸다. 2차원 배열은 다음과 같이 선언하면 된다.

● 배열 선언 형식 : 자료형 배열이름[행의 수][열의 수] ;

학생 4명의 번호와 컴퓨터 점수를 입력받으려고 한다. 그러면, 4명에 대해 한 사람씩 번호와 점수를 저장해야 한다. 알아보기 쉽게 표로 그려보면 4행 2열로 그릴 수 있다.

● 배열 변수 선언 : int data[4][2] ;

기억장소의 공간은 $4 \times 2 = 8$ 개의 int형이 연속해서 확보되고, 변수는 data[0][0]~data[3][1]까지 사용이 된다. 배열을 선언하면 <표 I .8.1>과 같이 변수들이 논리적으로 배치된다.

<표 I .8.1> 배열의 배치

번호	점수
data[0][0]	data[0][1]
data[1][0]	data[1][1]
data[2][0]	data[2][1]
data[3][0]	data[3][1]

<표 I .8.2> 실제 입력 내용

번호	점수
1	80
2	90
3	70
4	50

실제 메모리의 저장은 행단위로 다음과 같이 차례대로 기억된다.

1	80	2	90	3	70	4	50
data[0][0]	data[0][1]	data[1][0]	data[1][1]	data[2][0]	data[2][1]	data[3][0]	data[3][1]

2차원 배열 변수의 초기화 방법을 살펴보자.

```
int data[2][3] = {50, 60, 70, 80, 90, 100} ;
int data[2][3] = {{50, 60, 70}, {80, 90, 100}} ;
char ch[2][6] = {"Hello", "Good!"} ;
char ch[ ][6] = {{'H', 'e', 'l', 'l', 'o', '\0'}, {'G', 'o', 'o', 'd', '\0'}} ;
```

초기화에서 배열 크기를 정해주지 않으면 컴파일러가 자동으로 크기를 잡는다.

아래 예제는 번호와 컴퓨터 점수를 입력받아 점수가 가장 높은 학생의 번호와 점수를 출력하는 프로그램이다.(단, 동점은 없고, 번호가 0이면 마친다.)

```
#include <stdio.h>
void main()
{
    int data[1000][2] ;
    int i, num=1, max=0 ;
    for(i=0; ; i++){
        scanf("%d %d", &data[i][0], &data[i][1]) ;
        if(data[i][0]==0) break ;
        if(data[i][1]>max) {
            max=data[i][1] ;
            num=data[i][0] ;
        }
    }
    printf("번호 : %d    점수 : %d \n", num, max) ;
}
```



문제 I .8.2

다음은 어떤 프로그램의 일부이다. 다음 프로그램을 실행시킨 후 s가 가질 수 있는 최대값은? (2007 중등)

```
a = 0 ;    b = N ;    s = 0 ;
while ( a >= 0 && a <= N && b >= 0 )
    if ( c[a][b] > 0 ) {
        a++ ;    s++ ;
    }
    else {
        a-- ;    b-- ;    s++ ;
    }
}
```

- ① N ② $2 \times N$ ③ $2 \times N + 1$ ④ $3 \times N$ ⑤ $3 \times N + 1$

<풀이> ⑤



연습문제

1

하나의 문장(2자 이상)를 입력받아 그 단어의 철자를 반대로 나열하는 프로그램을 작성하시오.(단, 공백은 없앤다)
 입력) Oh! My god! 출력) !dogyM!hO

2

다음은 어떤 프로그램의 일부이다. 배열 a가 아래와 같을 때 다음 프로그램을 실행시킨 후 j값으로 알맞은 것은? (2007 초등)

a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
15	7	3	20	24	6	12	30	9	15

```

N = 10 ;   i = 1 ;   j = N + 1 ;
do {
    do i++ ;
    while ( a[i] <= a[1] ) ;
    do j-- ;
    while ( a[j] >= a[1] ) ;
    if ( i < j )
    {
        temp = a[i] ;   a[i] = a[j] ;   a[j] = temp ;
    }
} while ( i < j ) ;

```

- ① 5 ② 6 ③ 7 ④ 8 ⑤ 9

3

어떤 프로그램의 일부이다. 프로그램 실행 후 a[5]가 갖는 결과값은?

```

a[1] = 2 ;   a[2] = 3 ;   a[3] = 4 ;   a[4] = 5 ;   a[5] = 1 ;
for ( i = 1; i <= 5; i++ ) a[i] = a[a[i]] ;

```

- ① 1 ② 2 ③ 3 ④ 4 ⑤ 5

4

a[1]부터 a[10]에는 1부터 10까지의 서로 다른 수가 임의의 순서로 들어 있다. 다음 프로그램은 어떤 프로그램의 일부이다. 이 부분을 실행시켰을 때 k가 될 수 있는 값 중 최소값은? (2006 중등)

```
m = a[1] ;
n = a[2] ;
for ( i = 3; i < 10; i += 2 )
{
    j = i + 1 ;
    if ( a[i] > m ) m = a[i] ;
    if ( a[j] < n ) n = a[j] ;
}
k = m - n ;
```

- ① -5 ② -1 ③ 1 ④ 5 ⑤ 9

5

다음 프로그램을 실행시켰을 때 출력되는 수는 몇 개인가?(2005 교원)

```
for ( i = 1; i <= 10; i++ ) w[i] = 0 ;
for ( i = 1; i <= 10; i++ )
{
    j = i ;
    while ( j <= 10 )
    {
        w[j] = 1 - w[j] ;
        j += i ;
    }
}
for ( i = 1; i <= 10; i++ ) if ( w[i] == 1 ) printf( "%d\n", i ) ;
```

- ① 1개 ② 2개 ③ 3개 ④ 4개 ⑤ 5개



다음 프로그램의 실행 결과는? (2007 교원)

```
#include<stdio.h>
void main(void)
{
    int t=0, n=9, m, i, s=3, c=0 ;
    int d[10] ;
    for(i=0; i<=9; i++)
        d[i] = i+1 ;
    while(t <= n) {
        m = (t+n)/2 ;
        if(d[m] < s) { t = m+1; c++; }
        else if(d[m] > s) { n = m-1; c++; }
        else {
            printf("%d-%d", ++c, m) ;
            break ;
        }
    }
}
```

① 2-2

② 2-3

③ 3-2

④ 3-3

⑤ 4-3



1부터 10 사이의 숫자가 적혀진 다섯 장의 카드가 주어진다. 그 중 세 장의 카드를 골라 그 합의 일의 자리수를 가장 크게 만들려고 한다. 예를 들어 다섯 장의 카드가 (7, 5, 5, 4, 9)인 경우 (7, 4, 9)를 선택하면 합이 20이 되어 일의 자리수는 0이 되고, (5, 5, 9)를 선택하면 합이 19가 되어 일의 자리수는 9가 된다. 세 장의 카드를 뽑아 그 합의 일의 자리수를 9보다 크게 만들 수는 없으므로 구하고자 하는 답은 9가 된다. 다섯 장의 카드 번호가 담긴 배열 $a[1] \sim a[5]$ 가 주어질 때, 이와 같이 구한 일의 자리수의 최대값 m을 출력하는 프로그램을 작성하시오. (2006 중등)

입력파일(INPUT.TXT)

7 5 5 4 9

출력파일(OUTPUT.TXT)

9

8

다음은 오름차순으로 정렬되어 있는 두 배열 a와 b를 합쳐 오름차순으로 정렬된 하나의 배열 c를 만드는 프로그램의 일부이다. 배열 a에는 M개의 수가 a[1]부터 a[M]까지, 배열 b에는 N개의 수가 b[1]부터 b[N]까지 들어 있고, MAX는 이 모든 수보다 더 큰 수라고 할 때 빈 칸에 들어갈 내용으로 가장 알맞은 것은? (2006 초등)

```
p = q = 1 ;      r = 0 ;
a[M + 1] = b[N + 1] = MAX ;
while ( [ ] ) {
    r++ ;
    if ( a[p] < b[q] ) { c[r] = a[p]; p++; }
    else { c[r] = b[q]; q++; }
}
```

- ① $p < M \quad | \quad q < N$ ② $p < M \quad \&\& \quad q < N$ ③ $p \leq M \quad \&\& \quad q \leq N$
 ④ $p \leq M \quad | \quad q \leq N$ ⑤ $p + q \leq M + N$

9

문자 학점을 숫자 학점으로 바꾸는 프로그램을 작성하시오. 문자 학점은 A, B, C, D 및 F가 있다. 그리고 F를 제외한 문자 학점 다음에는 “+”가 나올 수 있다. 문자 학점 A, B, C, D 그리고 F에 대한 숫자 학점은 4, 3, 2, 1과 0이다. “+”는 숫자 학점을 0.5 증가시킨다. 예를 들어서 “B+”는 3.5가 된다. 실행 파일은 PR1.EXE라 한다. (2005 교원)

- <조건 1> 입력 시, 문자 학점에 해당되는 A+, A, B+, B, C+, C, D+, D, F 이외의 학점에 대해서는 “0(숫자)”를 출력한다.(단, 대소문자를 구분하지 않는다. 예를 들어, ”a”와 ”A”는 같이 처리한다.)
 <입력 형식> 사용자로부터 문자 학점을 입력 받는다.
 <출력 형식> 출력 파일은 OUTPUT.TXT라 한다. 이 파일에는 문자 학점에 해당되는 숫자 학점이 출력된다. 예를 들어 사용자 입력에서 “B+”이 입력되면 그 결과는 “3.5”가 출력되어야 한다.

입력파일(INPUT.TXT)

B+

출력파일(OUTPUT.TXT)

3.5



정렬되지 않은 학생들의 임의의 점수를 입력하여 석차를 계산하는 프로그램을 작성하시오. 점수는 동점이 있을 수 있으며, 이러한 경우 같은 석차로 처리한다. 예를 들어 5명의 점수 100, 90, 76, 60, 90이 입력되었다면 석차는 2등이 2명이고 3등은 없다. (단, 점수가 가장 높은 학생을 1등으로 한다. 실행파일은 “PR2.EXE”이다.) (2006 교원)

<입력형식> 처리할 점수의 개수와 처리할 점수 데이터는 “INPUT.TXT” 파일에서 입력받는다.

- 1) 입력 파일의 첫 번째 줄은 처리할 점수의 개수
- 2) 입력 파일의 두 번째 줄은 처리할 점수 데이터
(단, 각각의 점수는 빈칸으로 구별)

<출력형식> 석차를 계산한 후 점수와 석차를 “OUTPUT.TXT” 파일로 출력한다.

입력파일(INPUT.TXT)

10
60 34 55 100 90 76 60 90 80 87

출력파일(OUTPUT.TXT)

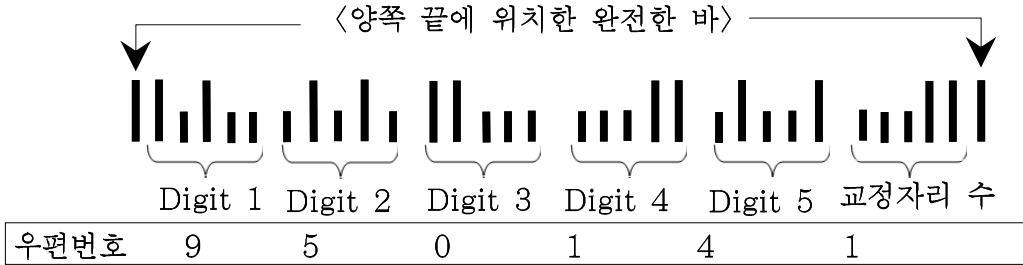
60	7
34	10
55	9
100	1
90	2
76	6
60	7
90	2
80	5
87	4



다량의 우편물을 보낼 때 우체국은 다량의 편지들을 빠르게 정렬하기 위하여 우편번호를 나타내는 바코드(Bar Code)의 사용을 장려한다. 이 우편번호 바코드는 바코드화된 다섯 자리 숫자 다음에 교정 자리수에 대한 인코딩이 위치하고, 양쪽 끝에 완전한 바가 추가된다. 교정 자리수는 다음과 같이 계산한다.

교정 자리수는 우편번호의 모든 자리수를 더한 합을 10의 배수로 만드는 수이다. 예를 들어, 우편 번호 95014는 자리수들의 합이 19이므로 이 합을 20으로 만들기 위한 교정 자리수는 1이다.

예)



우편 번호의 각 자리수와 교정 자리수는 다음 표에 의하여 인코딩 된다.

	7	4	2	1	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

표에서, 0은 높이가 1/2인 바를 나타내고 1은 완전한 바를 나타낸다. 자리수의 인코딩은 2개의 완전한 바와 3개의 높이가 1/2인 바의 조합이다. 바코드에 대응하는 자리수는 열의 가중치 7, 4, 2, 1, 0을 이용하여 쉽게 계산할 수 있다. 예를 들어, 01100은 $0 \times 7 + 1 \times 4 + 1 \times 2 + 0 \times 1 + 0 \times 0 = 6$ 이 된다. (단, 0은 예외이다)

우편번호를 입력하여, 바코드를 인쇄하는 프로그램을 작성하시오. 높이가 1/2인 바는 :로 나타내고 완전한 바는 |로 나타낸다. 실행파일은 PR2.EXE라 한다. (2005 교원)

〈입력 형식〉 사용자로부터 우편번호에 해당되는 숫자(5자리)를 입력 받는다.

입력에는 전혀 오류가 없다고 가정한다.

〈출력 형식〉 출력파일은 OUTPUT.TXT라 한다. 사용자가 입력한 숫자(5자리)에 해당되는 결과 값이 출력되어야 한다. 즉, 입력이 95014의 경우 코드로 변환된 ||:|:::|:||:||||:||:|:::|||이 출력되어야 한다.

〈입출력 예〉

입력파일(INPUT.TXT)
95014

출력파일(OUTPUT.TXT)

||:|:::|:||:||||:||:|:::|||



풀

이



```
#include<stdio.h>
#include<string.h>
void main()
{
    int i, len ;
    char data[1000] ;
    printf("문자열 입력 : ") ;
    gets(data) ;
    len=strlen(data)-1 ;
    for(i=len ; i>=0 ; i--)
        if(data[i]!=' ') printf("%c", data[i]) ;
}
```



②



③



③



③



③



③

```
#include <stdio.h>
void main()
{
    int i, j, p, m=0, s=0, a[6] ;
    FILE *fp1, *fp2 ;
    fp1=fopen("input.txt", "r") ;
    fp2=fopen("output.txt", "w") ;
```

```

for(i=1; i<=5; i++) fscanf(fp1, "%d", &a[i]) ;
for (i=1; i<=5; i++) s+= a[i] ;
for (i=1; i<=4; i++) {
    for (j=i+1; j<=5; j++) {
        p = ( s-a[i]-a[j] ) % 10 ;
        if ( p>m ) m=p ;
    }
}
fprintf(fp2, "%d\n", m) ;
}

```

⑧

⑨

```

#include <stdio.h>
#include <string.h>
void main()
{
    float result=0 ;
    char str[100] ;
    FILE *fp1, *fp2 ;
    fp1 = fopen("input.txt", "r") ;
    fp2 = fopen("output.txt", "w") ;
    fscanf(fp1, "%s", str) ;
    strcpy(str, strupr(str)) ;
    if(str[1]=='\0' || str[2]=='\0') {
        switch (str[0]) {
            case 'A' : result=4 ; break ;
            case 'B' : result=3 ; break ;
            case 'C' : result=2 ; break ;
            case 'D' : result=1 ; break ;
            default : result=0 ;
        }
        if(str[1]=='+') result+=0.5 ;
    }
    printf(fp2, "%.1f", result) ;
}

```

10

```
#include <stdio.h>
int score[1000], rank[1000] ;
void main()
{
    int i, j, n ;
    FILE *in=fopen("input.txt", "r") ;
    FILE *out=fopen("output.txt", "w") ;
    fscanf(in, "%d", &n);
    for(i=0; i<n; i++) fscanf(in, "%d", &score[i]) ;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            if(score[i]<score[j]) rank[i]++;
    for(i=0; i<n; i++)
        fprintf(out, "%-6d%-5d \n", score[i], rank[i]+1) ;
}
```

11

```
#include<stdio.h>
int data, bar[7] ;
int lain[10][5]={ {1,1,0,0,0}, {0,0,0,1,1}, {0,0,1,0,1},
                  {0,0,1,1,0}, {0,1,0,0,1}, {0,1,0,1,0},
                  {0,1,1,0,0}, {1,0,0,0,1}, {1,0,0,1,0},
                  {1,0,1,0,0} } ;
FILE *fp ;
void process(void) ;
void output(void) ;
void main()
{
    fp= fopen("input.txt", "r") ;
    fscanf(fp, "%d", &data) ;
    fclose(fp) ;
    process() ;
    output() ;
}
```

```
void process(void)
{
    int i, hap=0 ;
    for(i=0; i<5; i++) {
        bar[4-i] =data%10 ;
        data=data/10 ;
        hap+=bar[4-i] ;
    }
    bar[5]=(hap/10+1)*10-hap ;
    printf("b[5]= %d \n", bar[5]) ;
}
void output(void)
{
    int i, j ;
    fp= fopen("output.txt", "w") ;
    fprintf(fp, "%c", '|') ;
    for(i=0; i<=5; i++) {
        for(j=0; j<=4; j++) {
            if(lain[bar[i]][j]==0) fprintf(fp, "%c", '!) ;
            else fprintf(fp, "%c", '|') ;
        }
    }
    fprintf(fp, "%c \n", '|') ;
}
```

9. 포인터

포인터 변수는 데이터값을 저장하고 있는 보통의 변수와는 다르게 데이터 값은 가지고 있는 메모리 주소를 저장하고 있다. 포인터를 이용함으로써 기억장소를 효율적 접근하고 메모리의 데이터에 쉽게 접근할 수 있다. 종류는 **번지 연산자 &와 간접번지 연산자 ***가 있다. 두 연산자 모두 연산자 뒤에 하나의 피연산자만 필요로 하는 단항 연산자이므로 곱하기 *와 AND 연산자 &와는 구분이 될 것이다. 포인터 연산자에 대해 살펴보자.

가. 주소 연산자(&)

주소 연산자로서 해당 변수의 주소를 나타낸다.

● 형식 : & 변수 //변수의 주소를 의미

다음은 변수 a의 값과 a의 메모리 주소를 출력하는 프로그램이다.

```
#include <stdio.h>
void main()
{
    int a=10;                                // %u는 메모리 주소를 출력
    printf("a= %d  저장된 주소 : %u", a, &a);
}
```

특히, 주소 연산자(&)는 scanf()문에서 많이 볼 수 있다.

나. 간접 번지 연산자 (*)

간접 번지 연산자(*)는 포인터 앞에 붙여서 포인터 변수가 가리키는 메모리 주소에 기억되어 있는 내용을 나타내는 것이다. 즉, 저장되어 있는 번지에 가면 결과 값이 있다는 뜻이다. 여기서 포인터(pointer) 변수라는 것은 변수의 실제 데이터가 저장되어 있는 메모리 주소를 저장하고 있는 변수이다. 결국, 포인터 변수를 찾아서 다시 저장되어 주소를 찾아가야만 실제 결과 값을 찾을 수 있는 것이다. 포인터 변수의 선언은 다음과 같다.

● 선언 형식 : 자료형 *포인터 변수 ;

● 선언 이후의 사용 방법 2가지

포인터 변수 // 번지를 저장할 수 있는 변수
***포인터 변수** // 포인터 변수의 내용이 가리키는 값

다음 예를 살펴보자

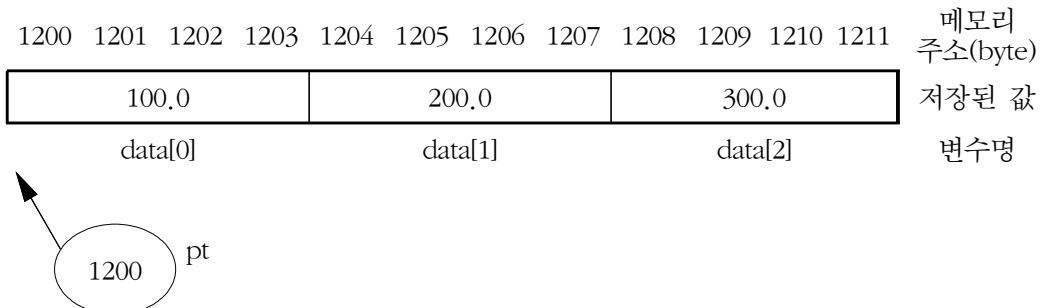
```
#include <stdio.h>
void main()
{
    char a, ch[6], *pt ;           // 포인터 변수 선언
    a= 'K' ;
    ch = "Hello" ;
    pt = ch ;                   // 배열명 ch는 배열의 시작 주소
    printf(" a = %c    pt = %s    *pt = %c    ch = %s \n", a, pt, *pt, ch) ;
}
```

실행 결과	a =K pt = Hello *pt = H ch = Hello
-------	---

다. 배열과 포인터 관계

다음과 같이 배열이 선언되었을 때 배열과 포인터의 관계를 살펴보도록 하자.
 실수형(4byte를 사용)으로 data라는 배열을 선언하고 1200번지부터 메모리에
 선언되었다고 하자.

```
float data[3]={100.0, 200.0, 300.0}, *pt ;
pt = data ;
```



포인터와 배열의 관계는 <표 I .9.1>와 같이 요약할 수 있다.

〈표 I .9.1〉 포인터와 배열의 관계

내 용	같은 표현들	
	기본적으로 동일한 표현	pt = data; 했을 경우
시작변지1200	data &data[0]	pt
data[2]의 값 300.0	data[2] *(data+2)	*(pt+2)

data[2]==*(data+2)이므로 배열에서 괄호[] 한 개는 *()와 바꿀 수 있다는 것을 알 수 있다. k[5][3]이 있다면 k[5][3]의 똑같은 의미의 다른 표현은 *(k[5]+3)과 *(*(k+5)+3)이 될 수 있다.

다음의 예를 통해 배열과 함수의 관계를 살펴보자.

```
#include <stdio.h>
void main()
{
    int num[4]={10, 20, 30, 40} ;
    char **pt ;
    char *str[]={("Flower", "Beautiful", "Happy")} ;
    pt=str ;
    printf("*num=%d \t\t *(num+2)=%d \n", *num, *(num+2)) ;
    printf("**pt=%c \t\t str[0][0]=%c \n", **pt, str[0][0]) ;
    printf("*str[1]=%s \t *(pt+1)=%s \n", str[1], *(pt+1)) ;
    printf("*(*(str+2)+2)=%c \t *(*(pt+2)+2)=%c \n",
          *(*(str+2)+3), *(*(pt+2)+3)) ;
}
```

실행 결과	*num=10 *(num+2)=30 **pt=F str[0][0]=F *str[1]=Beautiful *(pt+1)=Beautiful *(*(str+2)+2)=p *(*(pt+2)+2)=p
-------	---

연습문제

1

아래와 같이 선언이 되었을 때 printf() 함수 %s 기호를 사용하여 출력을 하고자 할 때 결과값이 같은 것만 모두 고른 것은?

```
char *ch[2]={"abcdefg", "hijklmn"} ;
```

- | | | | |
|----------|-----------|---------|-------------|
| Ⓐ ch | Ⓑ *ch | Ⓒ ch[0] | Ⓓ *(ch+0) |
| Ⓔ &ch[0] | Ⓕ ch[0]+1 | Ⓖ *ch+1 | Ⓗ &ch[0][0] |

- ① ⒶⒷⒸ ② ⒷⒸⒹ ③ ⒷⒸⒻⒸ ④ ⒷⒹⒺ ⑤ ⒷⒸⒹⒻⒸⒹ

2

한 문자열에서 찾고자 하는 영문자가 몇 개 있으며 위치는 어디인지를 알려주는 프로그램을 작성하시오.

입력파일(INPUT.TXT)

```
Thanks very march.  
a
```

출력파일(OUTPUT.TXT)

```
위치 : 3 14  
개수 : 2
```

3

3*3배열에 대문자를 랜덤으로 입력한다. swap() 함수를 정의하여 실행의 결과처럼 행과 열이 다음과 같이 바뀌게 만드시오.

랜덤으로 입력된 내용 예

```
A B C  
H I J  
X Y Z
```

출력파일(OUTPUT.TXT)

```
A H X  
B I Y  
C J Z
```



9개의 서로 다른 자연수가 주어질 때, 이들 중 최대값을 찾고 그 최대값이 몇 번째 수인지를 구하는 프로그램을 작성하시오. (2007 초등)
예를 들어, 서로 다른 9개의 자연수 3, 29, 38, 12, 57, 74, 40, 85, 61 이 주어지면 이들 중 최대값은 85이고, 이 값은 8번째 수이다.
실행파일의 이름은 AA.EXE로 하고, 프로그램의 실행시간은 1초를 넘을 수 없다. 부분 점수가 주어질 수 있다.

입력 파일의 이름은 INPUT.TXT로 한다. 첫째 줄부터 아홉
〈입력 형식〉 번째 줄까지 한 줄에 하나의 자연수가 주어진다. 주어지는 자연

수는 100보다 작다.

〈출력 형식〉 출력 파일의 이름은 OUTPUT.TXT로 한다. 첫째 줄에 최대값
을 출력하고, 둘째 줄에 최대값이 몇 번째 수인지를 출력한다.

〈입력과 출력의 예〉

입력파일(INPUT.TXT)

3
29
38
12
57
74
40
85
61

출력파일(OUTPUT.TXT)

85
8



풀



④ ઉદ્દરો

출력은 abcdefg가 된다.



```
#include<stdio.h>
#include<string.h>
void count() ;
void input() ;
char word, data[1000] ;
FILE *fp1, *fp2 ;
void main()
{
    input() ;
    count() ;
}
void input()
{
    fp1=fopen("input.txt", "r") ;
    fgets(data, 1000, fp1) ;
    fscanf(fp1, "%c", &word) ;
    fclose(fp1) ;
}
void count()
{
    int i=0, len, cnt=0 ;
    fp2=fopen("output.txt", "w") ;
    len=strlen(data) ;
    fprintf(fp2, "위치 : ") ;
    for(i=0; i<len; i++)
        if(data[i]==word) {
            cnt++ ;
            fprintf(fp2, "%d ", i+1) ;
        }
    fprintf(fp2, "\n개수 : %d \n", cnt);
    fclose(fp2) ;
}
```



swap()함수는 교환을 목적으로 사용자가 정의한 함수이다.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void input(char [] [3]) ;
void swap(char *, char *) ;
void main()
{
    int i, j ;
    char ch[3][3] ;
    FILE *fp= fopen("output.txt", "w") ;
    input(ch) ;
    for(i=0; i<3; i++) {
        for(j=0; j<3; j++) printf("%c ", ch[i][j]) ;
        printf("\n") ;
    }
    printf("\n") ;
    for(i=0; i<3; i++) {
        for(j=i; j<3; j++) swap(&ch[i][j], &ch[j][i]) ;
    }
    for(i=0; i<3; i++) {
        for(j=0; j<3; j++) fprintf(fp, "%c ", ch[i][j]) ;
        fprintf(fp, "\n") ;
    }
}

void input(char ch[][])
{
    int i, j ;
    srand(time(NULL)) ;
    for(i=0; i<3; i++)
        for(j=0; j<3; j++) ch[i][j] = rand()%26 + 65 ;
}

void swap(char *a, char * b)
{
    char temp ;
    temp = *a ;
    *a = *b ;
    *b = temp ;
}
```



```
#include <stdio.h>

void process() ;
void output() ;
FILE *fp ;
int max=0, num=0 ;

void main()
{
    process() ;
    output() ;
}

void process()
{
    int i, a ;
    FILE* fp = fopen("input.txt", "r") ;
    for (i=1; i<=9; i++) {
        fscanf(fp, "%d", &a) ;
        if (a>max) {
            max = a ;
            num = i ;
        }
    }
}

void output()
{
    FILE* fp = fopen("output.txt", "w") ;
    fprintf(fp, "%d \n %d \n", max, num) ;
    fclose(fp) ;
}
```


• 정보올림피아드 및 프로그래밍 교육 교재



Visual Basic

제 II 장에서는 Visual Basic 프로그래밍 기초에 관련된 내용을 살펴본다. 여기서는 정보올림피아드와 교원컴퓨터프로그래밍 경진대회에서 사용하는 Visual Basic 6.0의 툴 사용법과 Basic언어의 기본 문법, 기출문제 풀이 등을 수록하였다.

1. Visual Basic 6.0 시작하기

Visual Basic은 C언어에 비해 문법이 간단하고 프로그래밍이 쉽다는 이유로 정보올림피아드 대회와 교원프로그래밍 대회에서 많은 사람들이 활용하고 있다. 여기서는 Visual Basic 6.0의 실행법에 대해 살펴보고 기본적인 문법, 기출문제 풀이 등에 대해 살펴볼 것이다. 먼저 기본적인 Visual Basic 실행방법에 대해 살펴보자.

가. Visual Basic 6.0 실행법

모듈 프로그래밍을 하는 일반적인 순서를 살펴보면, [문제분석 및 설계] → [프로젝트 생성] → [폼 제거] → [모듈 추가] → [코딩] → [모듈 저장] → [프로젝트 저장] → [프로그램 실행(테스트)] → [오류 발생 시 디버깅(오류 수정)] → [프로그램 실행(테스트)] → [모듈 저장] → [실행 파일 만들기] → [파일 제출] 순으로 정리할 수 있다.

1) 폼을 제거한 후 모듈에서 코딩하기

보통, 자격증 시험이나 정보올림피아드 대회에서 사용하는 방법으로 폼을 제거하고 새로운 모듈을 추가하여 모듈에서 코딩을 하는 방법으로, 모니터로 출력하지 않고 파일에서 입력받아 처리 작업을 거쳐 결과를 파일로 출력하는 과정을 거친다. 순서는 다음과 같다.

- ① [파일] → [새 프로젝트] 메뉴를 실행한다.
- ② 새 프로젝트 대화상자에서 [표준 EXE]을 선택한 후 확인 버튼을 누른다.
- ③ 이제 [프로젝트] → [Form1 제거] 메뉴를 실행하여 폼을 제거한다. 폼을 저장하겠는지 물어보면 저장안함을 누르면 된다. 폼을 제거하면 화면 중간에 폼 창이 없어질 것이다.
- ④ [프로젝트] → [모듈 추가] 메뉴를 실행한다. 모듈 추가 대화상자가 나타나면 모듈을 선택한 후 열기 버튼을 누른다.
- ⑤ [파일] → [프로젝트 저장] 메뉴를 실행한 뒤, 저장 위치(PRE01 폴더를 만듦)를 정한 지정한 다음 파일 이름을 “PRE01”이라고 적고 저장 버튼을 눌러 프로젝트를 저장한다. 확장자는 “vbp”라는 것을 기억하자.

- ⑥ 만약, “SourceSafe를 이 프로젝트에 삽입시키겠는가?”라는 문구의 **【소스 코드
컨트롤】** 대화상자가 뜨면 “아니요(No)”를 누른다.
- ⑦ **[파일] → [Module1 저장(Ctrl+s)]** 메뉴를 실행시켜 모듈을 저장한다. 저장 위치는 프로젝트 파일과 동일한 곳으로 옮기고, 파일 이름을 “PRE01”로 바꾸고 저장 버튼을 누른다. 확장자가 “BAS”라는 것을 기억하자.
- ⑧ 이제 **코딩**을 해야 한다. 즉 프로그램을 작성해야 하는데, 출력을 어떻게 하느냐에 따라 프로그램의 출력문이 조금씩 달라지기 때문에 여기에 대해서 잠시 살펴보자. 대회에선 반드시 파일로 입출력을 해야 한다. 그 외에 단순히 확인만 하려고 할 때는 직접 실행창을 열어서 결과를 확인하는 방법, 메시지 박스(MessageBox)에 실행 결과를 출력하는 방법이 있을 수 있다. 지금부터는 파일로 출력, 직접 실행창으로 출력, 메시지박스로 출력의 3가지 방법에 대해 각각 알아보자.

〈표 II.1.1〉 출력 방법

출력 방법	관련 명령어	사용 방법
(1) 파일로 출력	open 파일이름 Mode as 파일번호 print 파일번호 문자열 close 파일번호	파일을 열어서 확인을 해야 함 (★ 대회에서는 반드시 파일로 결과를 출력해야 함 ★)
(2) 직접 실행 창	debug.print 문자열	보기-> 직접 실행창 실행 (단축키 : Ctrl+G)
(3) 메시지박스	MsgBox 문자열	메시지 박스가 새로 나타남

가) 첫번째 방법 : 파일로 입출력

파일로 입출력하는 순서는 다음과 같다.

- ① 다음과 같이 코드 입력창에 프로그램을 작성한다.

```
Sub Main()
    Open "output01.txt" For Output As #1
    Print #1, "Good morning!"
    Close #1
End Sub
```

- ② 도구모음의 [저장]도구를 누르거나 [파일] → [PRE01.bas 저장]를 실행 시킨다.
- ③ [실행] → [시작 메뉴(F5)] 실행 또는 도구모음의 [시작]도구  를 누른다.
- ④ 오류가 없으면 파일에 내용이 출력되었을 것이다. 하지만, 오류가 발생하면 오류 메시지 상자가 나타나 컴파일 오류가 발생했다고 알려준다. 확인 버튼을 누른 후 오류를 수정해 준다.
- ⑤ 정상적으로 실행이 완료되면, 모듈 파일이 저장된 폴더에 “output.txt” 텍스트 파일이 생성된다. 더블클릭하여 메모장에서 내용을 확인한다. 또는 Visual Basic 프로그램에서 파일을 불러오기 하여 볼 수도 있다.
- ⑥ 마지막으로 실행(EXE) 파일을 생성해야 한다. [파일] → [PRE01.exe 만들기] 메뉴를 실행시켜 실행 파일의 이름을 PRE01로 적고 확인 버튼을 누른다. 파일 생성 위치는 프로젝트 파일이나 모듈 파일과 동일한 곳으로 하면 된다.
- ⑦ 탐색기나 내컴퓨터로 파일들을 저장한 곳으로 이동하여 실행 파일을 확인한다.
- ⑧ 파일의 제출은 프로젝트 파일, 모듈 파일, 실행 파일, 출력 파일 4개를 제출하면 된다. 만약, 입력 파일도 존재하면 모두 5개의 파일을 제출해야 한다.

나) 두번째 방법 : 직접 실행 창으로 결과 출력하기

앞에서 설명한 프로젝트 파일 저장, 모듈 저장까지는 동일하다.

- ① [보기] → [직접 실행 창 (Ctrl+G)] 메뉴 실행하면 화면 하단에 직접 실행창이 나타난다.
- ② 코드 입력 창에 다음과 같이 소스를 입력한다.

```
Sub main()
    Debug.Print "Hello!"
    Debug.Print "경상남도교육청입니다."
End Sub
```

Debug.Print “Hello!” : 디버깅 원도우(직접실행 창)에 “Hello!”를 출력 한다.

- ③ [파일] → [PRE01.bas 저장]을 누르거나 도구모음의 저장 도구를 눌러 모듈을 저장한다.
- ④ [실행] → [시작] 메뉴를 실행시킨다. 결과는 직접 실행 창에서 볼 수 있다.

다) 세번째 방법 : 메시지 박스를 이용한 출력

앞에서 설명한 프로젝트 파일 저장, 모듈 저장까지는 모두 동일하다.

- ① 코드 입력 창에 다음과 같이 코딩한다.

```
Sub main()
    MsgBox ("hello!" + vbCrLf + "경상남도교육청입니다.")
End Sub
```

- ② [파일] → [PRE01.bas 저장]을 누르거나 도구모음의 저장 도구를 눌러 모듈을 저장한다.
- ③ [실행] → [시작] 메뉴를 실행시킨다. 그러면 메시지박스가 나타나고, 확인 버튼을 누르면 메시지박스가 사라진다.

교재에서의 입출력 방법

이 교재에서는 파일 입출력에 대해 공부하기 이전의 문법에 대해서는 편의상 MsgBox문을 이용한 메시지박스로의 출력이나 Debug.Print문을 이용한 직접 실행 창으로 출력하는 방법들을 다루게 될 것이며, 입력은 InputBox문을 이용하여 입력상자에서 입력을 받도록 할 것이다.

2. 자료형과 연산자

상수는 변하지 않는 값을 상수(constant)라 하고, 변수(variable)는 하나의 기억 장소로서 값이 변할 수 있는 것이다. 여기서는 상수와 변수의 자료형과 사용법, 연산자들의 종류 등에 대해 살펴볼 것이다

가. 상수

상수란 한 프로그램 내에서는 변경되지 않는 고유한 값으로, 숫자나 문자, 문자열, 날짜 등이 모두 상수가 될 수 있다. Visual Basic에서는 내부 또는 시스템 정의 상수와 사용자 정의 상수가 존재하며, 사용자 정의 상수를 사용할 때는 기억하기 쉬운 이름으로 작성해야 하며, 상수는 코드의 어느 곳에서나 실제 값을 대신하여 사용할 수 있다

1) 사용자 정의 상수의 선언 : Const 사용

◎ 형식 : [Public | Private] Const 상수이름 [As 자료형] = 상수값

[Public | Private]은 선택사항이고, Public은 ‘공공의’라는 의미로 외부에서도 사용할 수도 있다는 의미이다. Private은 ‘개인적인’의 의미로 선언을 한 모듈이나 함수 등의 범위 내에서만 사용할 수 있다는 의미이다.

Const PI = 3.14159

' 상수 선언

Const 작성일 = #10/10/2007#

' 날짜 상수 값은 #~# 묶어야 함

나. 변수

변수란 데이터가 저장될 수 있는 기억 장소의 이름으로, 상수와 달리 변수의 내용은 계속해서 바뀔 수도 있는 것이다. 변수는 자신이 저장할 수 있는 내용의 자료형을 갖고 있으며 보통 Dim문으로 변수 선언을 할 때 부여한다. 변수 선언을 하지 않으면 자료형은 상황에 맞게 자동으로 부여된다.

1) 변수의 자료형

자료형(data type)은 데이터의 표현 형식을 말하는 것으로, 변수에 자료형이 지정되면 변수에 저장될 수 있는 값의 형태와 변수가 차지하는 기억장소의 크기도 정해지게 된다. 자료형의 종류는 <표Ⅱ.2.1>과 같다.

<표 II.2.1> 자료형

자료형	크기(Byte)	형선언 문자	표현 범위	간단한 설명
Boolean	1	없음	True(1) 또는 False(0)	참, 거짓
Byte	2	없음	0 ~ 255	부호가 없는 정수
Integer	2	%	- 3 2 7 6 8 (- 2 ¹⁵) ~ 32767(2 ¹⁵ -1)	부호가 있는 정수
Long	4	&	- 2 , 1 4 7 , 4 8 3 , 6 4 8 ~ 2,147,483,647	보호가 있는 정수
Single (단정도 실수)	4	!	음수 : - 3 . 4 0 2 8 2 3 E 3 8 ~ -1.401298E-45 0 양수 : 1.401298E-45 ~ 3.402823E38	4byte의 부동 소수점
Double (배정도 실수)	8	#	음수값 : -1.79769313486232E-308 ~ -4.904665645841247E-324 0 양수값 : 4.904665645841247E-324 ~ 1.79769313486232E-308	8byte의 부동 소수점
Currency	8	@	-922,337,203,685,477.5808 ~ 922,337,203,685,477.5807	통화 형
Date	8	없음	100년 1월 1일	날짜 형
Object	4	없음	모든 객체 참조	
String (고정길이)	문자열 길이	\$	0 ~ 약 65,400 바이트	고정 크기 문자열
String (가변길이)	문자열 길이 + 10	없음	0 ~ 약 2조 바이트	가변 크기 문자열
Variant (문자)	문자열 길이 + 22	없음	String(가변길이)와 비슷함	자료형이 선언되지 않을 경우의 기본 자료형
Variant (숫자)	16	없음	Double 형 범위와 같음	"
사용자 정의 형식				사용자가 새로 정의

가) 정수 자료형(Byte, Integer, Long)

소수점을 포함하지 않은 정수형을 저장할 수 있는 자료형으로, 종류는 Byte, Integer와 Long 등 3가지가 있다. 변수의 선언은『Dim 변수명 As 자료형』 형식으로 선언할 수 있다. 정수형의 변수 선언의 예를 살펴보자.

● 정수 자료형 변수 선언 예

```
Dim K As Integer
```

나) 실수 자료형 (Single, Double)

소수점을 포함할 수 있는 실수를 저장할 수 있는 자료형이다. 소수점 표현, 지수 표현 등의 아주 큰 수나 아주 작은 수를 표현할 수 있다. 종류로는 4바이트 크기의 Single형, 8바이트 크기의 Double형 2가지가 있다.

● 실수 자료형 변수 선언 예

```
Dim sum As Single
```

다) 문자열 자료형

문자열(String)은 큰 따옴표(" ") 안에 문자들을 일렬로 나타낸 것으로, 이러한 문자열을 저장할 수 있는 자료형을 문자열 자료형이라 한다. 문자열 자료형은 선언할 때 크기를 정해주면 고정 길이로 사용하는 것이고, 정해주지 않으면 가변 길이로 사용하는 것이다.

● 문자열 자료형의 변수 선언 예

```
Dim st1 As String * 10
```

' 고정 길이의 문자열 변수 선언 : 크기 10바이트

```
Dim st2 As String
```

' 가변 길이의 문자열 변수 선언 : 결과값 대입될 때 정해짐

```
st1 = "abvcdeghijklmn"
```

' 정해진 크기보다 입력되는 문자열이 길므로 "klmn"은 잘린다.

```
st2 = "abcd"
```

' st2의 크기는 4가 된다.

다음의 문자열과 관련된 예제를 살펴보자.

```
Option Explicit
Sub main()
    Dim st1 As String          ' 가변길이 문자열 변수 선언
    Dim st2 As String
    Dim st3 As String * 10      ' 고정길이 문자열 변수 선언
    Dim imsi As Integer
    st1 = "B"
    st2 = Chr(Asc(st1) - 1)     ' Chr()함수는 아스키값을 문자값으로 변환
    st3 = "ABCDEFGHIJKLMN"      ' st3변수의 크기보다 큰 부분은 잘립
    imsi = Asc("A") - Asc("a")   ' Asc()함수는 아스키 코드값으로 변환
                                ' _는 다음줄에 연속해서 적는다는 의미
    MsgBox "st1 = " & st1 & " st2 = " & st2 & _
           vbCrLf & "imsi = " & imsi & _
           vbCrLf & "st3 = " & " " & st3
End Sub
```

실행 결과	st1 = B st2=A imsi = -32 st3 = ABCDEFGHIJ
-------	---

라) Variant 자료형

Visual Basic에서는 변수를 선언할 때 형식을 지정해 주지 않으면 자동으로 해당 변수를 Variant형식으로 지정하게 된다. Variant형에서 문자열을 저장할 경우에는 크기가 문자열의 크기 +22를 하여 형의 크기가 정해지고, 정수나 실수 같은 숫자의 경우는 16바이트로 형의 크기가 정해진다.

마) 논리 데이터 자료형

논리적인 값 참(True), 거짓(False)의 값을 표현한다. 일반적으로 선언될 때 기본적인 값은 거짓이다. 변수 선언의 예는 다음과 같다.

```
Dim k As Boolean
```

2) 변수 선언

변수 선언은 메모리상에 사용할 공간을 마련한다는 의미이다. Visual Basic에서는 변수를 선언하지 않고 바로 사용할 수도 있는데, 이 경우는 자료형이 Variant로 저장되는 값에 의해 크기가 정해진다. 이러한 변수 사용 방

법을 **묵시적 방법**이라고 한다. 반대로 사용할 변수명과 변수의 자료형을 선언하고 사용하는 방법을 **명시적인 방법**이라 한다. 먼저 변수 선언 방법부터 알아보자.

가) 선언 방법

변수 선언은 Dim문을 이용하여 선언 형식은 다음과 같다.

● 형식 :

Dim 변수명 As 자료형 ' 하나의 변수 선언

Dim 변수명1, 변수명2, 변수명3, … As 자료형

' 자료형이 같은 여러 가지 변수 선언

Dim 변수명1 As 자료형1, 변수명2 As 자료형2, 변수명3 As 자료형3, …

' 자료형이 다른 여러 개의 변수 선언. 콤마로서 구분함.

예) Dim sum As Long

Dim 합계, 평균, 등수 As Long

Dim A As Integer, L As Long, FT As Boolean

나) 명시적 변수 선언 방법(Option Explicit문 이용)

① 변수를 프로그램의 선언부에서 미리 선언한 후 사용해야 한다.

② Option Explicit문을 프로그램 시작부분에 입력한 후, Dim문으로 변수 선언해야 한다.

다음의 예제는 1부터 10까지의 합을 구하는 프로그램이다. 가장 먼저 Option Explicit문을 입력하고, 그 다음에 변수가 선언되어 있다.

<pre> Option Explicit Dim SUM As Integer , i As Integer Sub Main() For i = 1 To 10 SUM = SUM + i Next i MsgBox "1부터 10까지 합계 = " & SUM End Sub </pre>	' 명시적 선언 방법을 사용하겠다는 표시
--	------------------------

다) 묵시적 변수 선언 방법

① 변수를 선언하지 않고 사용하는 방법으로 Visual Basic은 자동으로 변수의 형을 Variant형으로 만든다.

다음 예는 뮤시적 변수 선언 방법으로 1부터 10까지의 합을 구하는 프로그램을 작성한 것이다.

```
Sub Main()
    For i = 1 To 10
        SUM = SUM + i
    Next i
    MsgBox "1부터 10까지 합계 = " & SUM
End Sub
```

다. 사용자 정의 자료형

C언어의 구조체와 비슷한 것으로, Visual Basic에서는 사용자 정의 자료형 또는 Type형이라고 말한다. 사용자 정의 자료형(Type형)은 서로 다른 자료형의 자료들을 하나의 집단으로 묶고 하나의 이름으로 관리하는 것을 말한다. 사용자 정의 자료형을 선언하는 방법은 다음과 같다.

● 선언 형식

[Private | Public] Type 사용자정의형 이름

구성요소 As 자료형

...

End Type

사용자정의형 이름은 Type문이 바로 변수로서 사용되는 것이 아니라, 여러 개의 구성 요소(필드)가 들어있는 사용자가 정의한 하나의 자료형으로 다음에 변수를 선언해야 한다.

Type형 변수의 각 구성요소들을 접근하고자 한다면 '.'연산자를 이용해야 한다. 예를 들면 temp.name = "홍길동"처럼 '.'연산자를 이용해야 한다.

다음은 Type문을 이용하여 한 사람(st1)에 대해 번호, 이름, 성적을 초기화 하는 예이다.

Option Explicit

```
Type Student          ' 사용자 정의 자료형 선언
    numb As Integer
    name As String * 10
    sungjuk As Integer
End Type

Sub Main()
    Dim st1 As Student      ' 변수 선언
    st1.numb = 35           ' data의 구성 요소에 값 저장
    st1.name = "홍길동"
    st1.sungjuk = 100
End Sub
```

라. 연산자

연산(operation)은 피연산자와 연산자에 의해 이루어진다. 수식 $a - b$ 는 연산이 되어지는 대상 a , b 의 피연산자(operand, 오퍼랜드)와, 빼기($-$)라는 연산 방법인 연산자(operator, 오퍼레이터)로 이루어진다. Visual Basic에서는 산술 연산자, 관계 연산자, 논리 연산자, 대입 연산자 등이 있다. 이런 연산자들 사이에는 우선순위도 있다. 연산자의 종류에 대해 살펴보자.

1) 산술 연산자

2개의 연산자를 필요로 하는 산술 연산자는 사칙연산($+$, $-$, $*$, $/$)과 지수 계산을 하는 $^$ 과 나머지를 나타내는 Mod 등이 있다.

다음 예는 산술 연산자에 관한 다양한 출력이다.

Option Explicit

```
Sub main()
    Dim a As Integer, b As Integer
    Dim c As Long, d As Long
    a = 100
```

```

b = 51
c = 50
d = 4
Debug.Print " a + b * a Mod b = " & a + b * a Mod b
Debug.Print " c / d = " & c / d
Debug.Print " c ^ d = " & c ^ d
End Sub

```

실행 결과

실행 결과	a + b * a Mod b = 100 c / d = 12.5 c ^d = 6250000
--------------	---

Debug.Print 문은 직접 실행 창에 결과를 출력하는 것이다. Debug.Print문에서 출력할 변수들을 콤마로 구분을 하면 10칸 정도 공백을 주면서 출력을 하고, 세미콜론(;)으로 구분을 하면 1칸을 띄우고 출력한다. 그리고 &를 통해 문자열을 합쳐 출력을 하면 공백없이 연결하여 출력을 한다.

2) 관계 연산자(비교 연산자)

관계 연산자는 두개의 피연산자에 대해 대소 관계를 비교하는 것이다. 관계 연산의 결과는 참(True, 1) 또는 거짓(False, 0)이 출력된다. 종류로는 =, <>, >, Like(두 문자열 비교), Is(두 변수의 동일 여부 판단) 등이 있다.

3) 논리 연산자

논리 연산자는 여러 개의 조건들을 논리적인 연산에 의해 최종적으로 참(True, 1)인지, 거짓(False, 0)인지를 판별하는 것으로, 연산자의 종류는 논리곱(And), 논리합(Or), 논리부정(Not) 등이 있다. 다음 <표Ⅱ.2.2>에서 연산자의 종류에 대해 알아보자.

〈표 II.2.2〉 논리 연산자

연산자	설명	사용법	사용 예 (a=5,b=3,c=10)	사용 예의 결과
And	논리곱 : 모두 참일 때 참이 됨	A And B	a>b And a>c	False
Or	논리합 : 하나라도 참이면 참이 됨	A Or B	a>b Or a>c	True
Xor	배타적 논리합 : 서로 결과가 다르면 참이 되고, 같으면 거짓이 됨	A Xor B	a>b Xor a>c	True
Not	논리부정 : Not 다음 피연산자의 논리를 부정함.	A Not B	Not (a>c)	True
그 외에 Eqv(등가), Imp(논리적 내포) 등이 있다.				

4) 비트 연산자

비트는 0과 1 값을 갖는 최소의 단위를 말한다. 좌변과 우변의 비트 값을 따라 And, Or, Not 등으로 결합되어 다음과 같은 논리 값을 얻을 수 있다. 다음 표에서 비트 연산자의 연산에 대해 살펴보자.

〈표 II.2.3〉 비트 연산자의 연산 진리표

A	B	A And B	A Or B	A Xor B	Not A	Not B	A Eqv B	A Imp B
0	0	0	0	0	1	1	1	1
0	1	0	1	1	1	0	0	1
1	0	0	1	1	0	1	0	0
1	1	1	1	0	0	0	1	1

5) 대입 연산자

오른쪽의 내용 혹은 결과 값을 왼쪽 변수에 대입(저장)하는 연산자를 대입 연산자(=)라고 한다.

6) 연결 연산자

두 개의 문자열을 연결하는 데 사용하는 연산자로서, “&”와 “+”가 있다. 여기서 문자열은 큰 따옴표(“ ”) 안에 포함된 문자들을 말한다.

〈표 II.2.4〉에서 연결 연산자의 종류에 대해 살펴보자.

〈표 II.2.4〉 연결 연산자의 종류

연산자	설명	사용 예 (a=100)	사용 예의 결과
&	두개의 문자열을 연결하는 연산자이다. 만약, 어느 한쪽이 숫자(정수, 실수)로 되어 있어도 문자열과 결합할 수 있다. 먼저 숫자를 문자열로 변환한 다음 문자열 연결을 수행한다.	"문자열" & "연결" "a = " & a	문자열 연결 a = 100
+	두개의 문자열을 연결하는 것으로 반드시 양 쪽 모두 문자열이어야 한다.	"문자열" + "연결"	문자열 연결

7) 연산자 우선순위

식에서 여러 개의 연산을 함께 수행할 경우 미리 정해진 연산자 우선순위에 따라 식의 각 부분을 해석하여 분해한다. 괄호 안의 연산은 항상 괄호 밖의 연산보다 먼저 수행되지만, 괄호 안에서는 일반적인 연산자 우선순위를 따른다. 산술 연산자, 비교 연산자, 논리 연산자의 우선순위를 요약하면 〈표 II.2.5〉 같다.

〈표 II.2.5〉 연산자의 우선순위

산술 연산자	비교 연산자	논리 연산자
지수(\cdot)	같다(=)	Not
부정(-)	같지 않다(<>)	And
곱셈과 나눗셈(*, /)	보다 작다(<)	Or
정수 나눗셈(\)	보다 크다(>)	Xor
나머지 연산(Mod)	보다 작거나 같다(<=)	Eqv
덧셈과 뺄셈(+, -)	보다 크거나 같다(>=)	Imp
문자열 연결(&)	Is	&
높다		낮다
\Rightarrow		\Rightarrow

& 연산자는 산술 연산자가 아니지만 우선순위는 모든 산술 연산자 뒤와 모든 비교 연산자 앞에 나온다.



문제 II.2.1

다음 연산에 대한 결과를 알아보자.

```
Option Explicit
Sub main()
    Dim A As Integer, B As Integer
    A = 100 : B = 10      ': 기호는 여러 문장을 한 줄에 입력할 경우 사용
    Debug.Print (A>100 And A<10) ;
    Debug.Print A Or B ;
    Debug.Print "결과 :" & (A + B) * A \ B ;
End Sub
```

<풀이>

실행 결과	False	110	결과 : 1100
-------	-------	-----	-----------



문제 II.5.2

수식 $\frac{100 \times 7^3}{4^3 - 3} = x$ 를 계산하여 x 의 결과를 출력하는 프로그램을 작성하시오. 단, 소수점 둘째자리에서 반올림하여 첫째자리까지 표현한다.

<풀이>

```
Sub main()
    Dim result As Double
    result = 100 * (7 ^ 3) / ((4 ^ 3) - 3)
    Debug.Print Round(result, 1);
End Sub
```

Round()함수는 정해진 자리수까지 표현하는 함수이다. 직접 실행창에서 결과를 볼 수 있다.

3. 파일 입출력

키보드나 모니터가 아닌 저장장치의 특정한 파일에서 입력 자료를 읽어오거나 출력 자료를 기록하는 일련의 과정들을 파일 입출력이라고 한다. 파일 입출력에 이용되는 파일에는 텍스트 파일(text file)과 이진 파일(binary file) 등이 있다. 프로그래밍 대회에선 반드시 입력 파일에서 입력 자료를 읽어오고 결과는 출력 파일에 기록한다. 그래서 소스 파일과 입출력 파일을 함께 결과물로 제출을 해야 한다. 보통 파일 입출력은 [파일열기] → [파일 읽고 쓰기] → [파일 닫기]의 과정을 거치게 된다. <표 II.3.1>에서 Visual Basic에서 다루는 파일을 알아보자.

<표 II.3.1> Visual Basic에서 다루는 파일

파일 종류	설명	처리 단위
순차 파일 (Sequential File)	데이터를 순차적으로 처리	행 단위
임의 파일 (Random File)	일정한 길이의 레코드 단위로 데이터를 입출력	레코드 단위
이진 파일 (Binary File)	바이트 단위로 섬세하게 제어하면서 입출력	바이트 단위

여러 대회에서는 사용하는 파일은 순차 파일이며, 메모장에서 텍스트 형식의 문서를 만들면 된다.

가. 텍스트 문서 만들기

1) 메모장에서 만들기

메모장에 입력 자료를 입력한 다음 저장한다. 저장 위치는 소스 파일이 있는 곳으로, 파일 형식은 텍스트 문서를 선택한 다음 저장하면 된다.

2) 순차파일

순차 파일은 파일의 내용을 처음부터 순차적으로 읽고 쓰기를 하는 파일 형식으로 행 단위로 파일 처리가 된다. 순차 파일은 텍스트 파일을 말하는 것으로 입력 파일을 만들거나 출력 파일의 결과를 메모장에서 쉽게 확인할 수 있다. 보통 대회에서는 순차 파일을 이용하므로 잘 익혀두자.

가) 파일 열기 : Open문

파일 열기는 Open문에 의해 이루어지는데, 형식은 다음과 같다.

Open "파일명" For 모드 As #파일번호 [Len=레코드 길이]

Open App.Path & "\파일명" For 모드 As #파일번호 [Len=레코드 길이]

예) Open "PRE01.txt" For Output As #2

출력 파일은 모듈이 저장된 폴더에 만들어지게 된다. 탐색기나 내 컴퓨터로 이동하여 열어보면 된다.

- ① 파일명 : 사용할 파일의 경로와 파일 이름이 들어가야 한다. 만약, 경로가 없으면 현재 폴더에 있는 파일을 의미한다. 현재의 위치를 알려주는 App.Path문은 정보올림피아드에서는 허용이 안 된다. 단지 c: 드라이브에 파일을 저장하여 연습할 때만 사용하자.
- ② 모드 : 파일의 사용목적을 지정하는 옵션으로 Input, Output, Append의 세 가지 모드가 있다.

〈표 II.3.2〉 파일 모드

모드 종류	설명
Input	순차 파일을 읽기 전용으로 열어 사용하는 모드
Output	순차 파일을 기록할 목적으로 열어 사용하는 모드 (기존 자료는 삭제)
Append	순차 파일을 해당 파일에 추가로 데이터를 기록하려고 할 때 사용

③ #파일번호 : 프로그래머가 정하는 것으로 파일에 대해 접근할 수 있도록 해 준다. 즉, 파일 입출력을 할 때 파일 이름을 사용하는 것이 아니라 파일번호를 이용한다.

④ Len=레코드 길이 : 파일에서 읽고 쓰기를 할 때 한번 수행에 사용되는 길이를 말한다. 예를 들어 아래와 같이 표현을 한다고 하자.

나) 파일 읽기 : Input

순차파일에서 문자열을 읽기 위해서는 Input #, Input(), Line Input # 등의 문장들을 사용해야 한다.

(1) Input

한 줄 단위로 읽기를 하되 변수 목록에 맞게 읽어 들이고자 할 때 사용 한다. 문자를 읽을 때는 한 줄 단위로 읽고, 숫자를 읽어 들일 때는 공백이나 다음 줄로 내리는 표시로 구분하여 읽어 들인다.

Input #파일번호, 변수 목록

예) Open "PRE01.txt" For Input As #1

Input #1, st1, st2

(2) Input() 함수

원하는 길이만큼 한꺼번에 데이터를 읽어 들일 때 사용하는 것으로, 읽어 들일 데이터 크기와 읽어 들일 파일을 지정하는 파일 번호가 주어져야 된다. 사용 형식은 다음과 같다.

변수 = Input(크기, #파일 번호)

또는 InputB(크기, #파일 번호)

예) Open "PRE01.txt" For Input As #1

st1 = Input(7, #1)

Input()문은 글자 수(크기)만큼 파일에서 읽어와 변수에 저장하는 것을 말하며, InputB()문은 바이트 크기만큼 파일에서 글자를 읽어온다. 단, 다음 줄로 내리는 라인피드(LF)문자나 공백도 글자의 수에 포함이 된다.

(3) Line Input

줄 단위로 읽어서 하나의 변수에 저장하는 것으로, 한 줄은 라인 피드(LF)나 캐리지 리턴(CR)이 나올 때까지를 이야기 한다. 라인 피드는 다음 줄로 내리는 문자이고, 캐리지 리턴은 현재 줄에서 맨 처음으로 옮겨주는 특수 문자이다.

Line Input #파일 번호, 변수 이름

예) Open "PRE01.txt" For Input As #1

Line Input #1, st1

다) 데이터 쓰기 : Print, Write

순차 파일에서 기록하는 명령어는 Print #, Write #문이 있다. Print #문은 표준 출력 명령어인 Print와 유사하다. 다만 파일로 출력을 한다는 점

이 다르다. Write # 문은 데이터 읽기를 할 때 사용하는 Input 문과 대응되는 명령어이다. 형식은 다음과 같다.

Print #파일번호, 출력 목록

Write #파일번호, 출력 목록

예) Open "output01.txt" For Output As #1

Print #1, "안녕하세요."

라) 파일 닫기 : Close

프로그램에서 열려진 파일이 더 이상 필요 없을 경우에는 파일을 다시 닫아야 한다. 파일 닫기는 Close문을 사용하는데, 형식은 다음과 같다.

Close #파일번호

마) 그 외의 관련 함수들

<표 II.3.3> 기타 함수들

종 류	설 명	사용 예
EOF	파일의 끝인지를 알려줌	EOF(#1)
LOF	Open문에 연 파일의 크기를 바이트 단위로 알려줌	Input(LOF(#1), #1)
KILL	디스크에서 파일을 삭제함	Kill 파일명

3) 임의 파일 처리

만약 일정한 길이의 레코드 단위로 데이터를 입출력하고자 한다면 임의파일 처리를 해야 한다. 임의파일은 파일 내의 특정한 위치로 직접 찾아갈 수 있다. 레코드 단위로 처리하기 위해서는 Type형으로 정의를 한 다음 처리해야 한다.

가) 파일 열기 : Open

Open명령어를 이용하여 파일을 열며, 모드를 Random(임의파일 접근의 용도)으로 하고, 레코드의 길이를 지정하여 파일의 읽고 쓸 때의 길이를 정 의해 주어야 한다. 형식은 다음과 같다.

Open 파일 이름 For Random As #파일번호 Len = 레코드 길이

나) 파일 읽기와 쓰기 : Get, Put

Get을 이용하여 레코드를 읽어 Type형 변수에 저장한다. Put은 Type형 변수의 값을 파일에 기록한다.

Get #파일번호, [레코드 번호], 변수

Put #파일번호, [레코드 번호], 변수

레코드 번호는 파일에서 읽거나 쓰기를 하고자 하는 레코드 번호이다. 만약, 생략되면 가장 최근의 다음 레코드를 가리킨다.

다) 파일 닫기 : Close

Close #파일번호

다음 텍스트 파일 "input.txt"에서 하나의 숫자와 하나의 문자열을 차례대로 읽어와 "output.txt" 파일에 출력하는 프로그램을 작성해 보자.

```
Option Explicit
Sub main()
    Dim num As Integer, str As String
    Open App.Path & "\input4.txt" For Input As #1      ' 입력파일 열기
    Open App.Path & "\output4.txt" For Output As #2   ' 출력파일 열기
    Input #1, num, str                                ' 입력파일에서 읽어오기
    Print #2, num & " " + str                         ' 출력파일 기록하기
    Close #1, #2                                       ' 열린 파일 닫기
End Sub
```

※ 대회에서는 App.Path & "\" 부분을 사용하면 안 됩니다. C:드라이버에 저장하여 결과 확인할 때만 사용해야 한다. Print문에서 "&"와 "+"기호는 문자열을 연결하는 기호이다. 다음부터는 App.Path문을 생략한다.

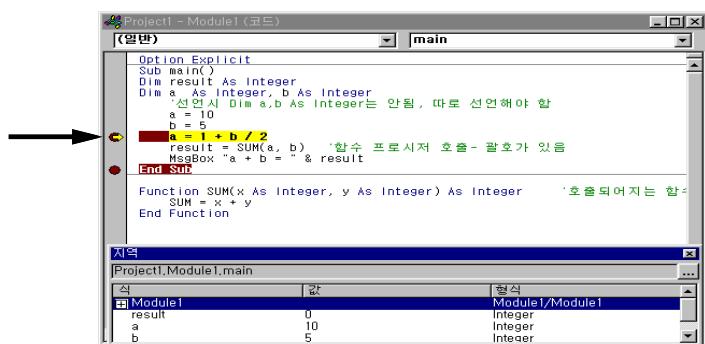
4. 디버깅

프로그램을 하다 보면 프로그램의 고장(failure)을 일으키는 에러(error) 또는 버그(bug)들을 찾게 되는데, 이런 에러를 찾아 올바르게 고치는 작업을 디버깅(debugging)이라고 한다. 에러는 문법적인 에러와 논리적인 에러가 있다. 문법적인 에러는 프로그래밍을 하면서 Visual Basic이 대부분 알려주거나, 도움말을 통해 쉽게 고칠 수 있다. 하지만, 논리적인 버그는 쉽게 찾을 수 없다. 그래서 Visual Basic에선 [디버그]메뉴를 제공하여 버그를 찾는데 도움을 준다.

가. 중단점 이용하기

프로그램에서 자신이 멈추고자 하는 위치에 중단점을 설정하고 프로그램을 실행시키면 프로그램이 실행을 하다가 중단점이 있는 곳에서 멈추고 디버깅 상태가 된다.

- ① 중단점의 설정 : 커서를 멈추고자 하는 라인에 두고, [디버그] → [중단점] 메뉴를 실행시킨다. 혹은 코드 입력창의 왼쪽 회색 여백을 클릭해도 된다.
- ② [보기] → [지역창] 메뉴를 실행시켜 지역창을 띄운다. 지역창에는 중단점에서의 변수들의 값을 나타낸다.
- ③ [실행] → [시작] 메뉴를 실행시킨다.
- ④ 중단점에서 실행이 멈추고, 중단점은 노란색으로 영역이 표시된다. 지역창을 통해 변수들의 값을 알아볼 수 있다. 다음 중단점으로 넘어가려면 [실행] → [시작] 버튼을 누르면 된다.



[그림 II.4.1] 중단점 사용 화면

- ⑤ 중단점의 해제는 [디버그] → [중단점 설정/해제]를 하거나 마우스로 왼쪽 여백의 중단점을 다시 클릭하면 해제된다. [모든 중단점 지우기]를 실행하여 모든 중단점을 제거할 수 있다.
- ⑥ [실행] → [종료] 메뉴를 실행시킨다.

나. 단계별 실행하기

실행을 이제 시작하거나 중단점에서 멈춰 있을 때 [디버그]-[한 단계씩 코드 실행] 메뉴를 선택한다. 그러면, 현재 위치에서 다음에 오는 문장을 하나씩 실행할 수 있다. 이 기능은 조건이나 분기문 등을 추적할 때 효율적으로 사용할 수 있다. 몇가지 실행 모드에 대해 살펴보자.

- ① 한 단계씩 코드 실행 : 현재 중단된 지점에서 다음 한 문장을 실행한다.
- ② 프로시저 단위 실행 : 다음에 오는 문장이 프로시저를 호출하는 문장일 경우 호출 프로시저를 실행하고, 호출문 다음 문장에서 실행이 중단된다. 프로시저 호출하는 문장이 아니면 다음 한 문장만을 수행한다.
- ③ 프로시저 나가기 : 현재 프로시저의 실행을 완료하고 현재 프로시저 외부의 다음 문장에서 실행이 중단된다.
- ④ 커서까지 실행 : 현재 커서가 있는 곳까지 실행을 하고 멈춘다.

다. 조사식 창 활용하기

변수나 표현식의 값에 대한 결과를 검사할 경우 사용할 수 있다.

- ① [보기] → [조사식 창] 메뉴를 실행시켜 조사식 창을 띄운다.
- ② 조사식에 추가하고 싶은 변수나 표현식을 마우스로 범위를 지정한다.
- ③ [디버그] → [조사식 추가] 메뉴를 실행하여 조사식에 추가한다.
- ④ 중단점을 추가한다.
- ⑤ [실행] → [시작] 메뉴를 눌러 실행시킨다.
- ⑥ 문장을 이동하면서 수행할 때마다 조사식이 바뀌는 것을 볼 수 있다.

조사식			
식	값	형식	컨텍스트
66 Sum(a, b)	9	Integer	Module1.main
66 a	4	Integer	Module1.main
66 result	9	Integer	Module1.main

[그림 II.4.2] 조사식 상자

라. 디버깅 상태에서 특정 변수의 값 검사

- ① 중단점을 지정한 후 실행을 시킨다.
- ② 디버깅 상태에서 변수의 값을 알고 싶은 변수를 마우스로 범위 선택 후 [디버그] → [간략한 조사식] 메뉴를 실행시킨다.(단축키: Shift 키 + F9)
- ③ 변수의 현재 값을 알 수 있다.



[그림 II.4.3] 간략한 조사식 상자



문제 II.4.1

중단점 설정 및 해제를 할 때 사용하면 편리한 단축키는 무엇인가?

- ① F5 ② Shift + F5 ③ F9 ④ Shift + F9 ⑤ F10

<풀이> ③



문제 II.4.2

특정한 변수를 계속해서 관리하려면 화면 아래에 어떤 창을 띄워두는 것이 좋은가?

- ① 직접 실행 창 ② 조사식 창 ③ 미리보기 창 ④ 지역창 ⑤ 속성창

<풀이> ②

5. 제어문

프로그램의 수행은 특별한 제어가 없다면, 일반적으로 처음부터 끝까지 아래쪽으로 순서대로 수행이 된다. 하지만 경우에 따라서는 흐름을 다른 곳으로 이동시켜 주거나 뒤로 되돌려 주는 등 다양한 방법으로 순서를 제어해야 하는데, 이런 역할을 하는 명령어들을 제어문이라고 한다. 제어문의 종류는 선택문, 반복문과 분기문 등이 있다.

가. 선택문

특정한 조건을 주고 선택을 하도록 하는 명령어를 선택문이라고 한다. If문, If~Else문, If~Elseif~Else문과 Select~case문 등이 있다.

1) If ~ Else문과 복합 If문

주어진 조건이 참이면 If문의 Then문 다음의 문장만 수행하고, 조건이 거짓이면 Else문 다음의 문장들만 수행하고 If문을 빠져 나가는 명령어이다. 기본 구조에서 End If 문장은 생략할 수 없다. 즉, 두 갈래의 길이 있어 처리하는 내용이 다른 것이다. 복합 If문은 If문 안에 또 다른 If문을 하나 이상 포함시키는 것을 말하며, 여러 개의 조건을 따지게 되는 것으로 선택의 방향이 훨씬 더 많은 것이다.

〈표 II.5.1〉 If~Else문과 복합 If문 기본 구조

If~Else문 기본 구조	복합 If문 기본 구조
If 조건 Then 문장 1 Else 문장2 End If 문장 3	If 조건1 Then 문장 1; Elseif 조건2 Then 문장 2; Else 문장 3; End If 문장 4;

다음은 숫자 n을 입력받아 n이 홀수인지 짝수인지 구분하는 프로그램이다.

```

Option Explicit
Sub main()
    Dim n As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n
    If (n Mod 2 = 0) Then
        Print #2, n & "은 짝수임"
    Else
        Print #2, n & "은 홀수임"
    End If
    Close #1, #2
End Sub

```

입력 내용	10	실행 결과	10은 짝수임
-------	----	-------	---------



문제 II.5.1

숫자 n을 입력 받아 3의 배수인지 2의 배수인지, 그 외의 숫자인지를 나타내는 프로그램을 작성하시오.

```

Option Explicit
Sub main()
    Dim n As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n
    If (n Mod 3 = 0 And n Mod 2 = 0) Then
        Print #2, "3의 배수이면서 2의 배수"
    ElseIf (n Mod 3 = 0 And n Mod 2 <> 0) Then
        Print #2, "3의 배수"
    ElseIf (n Mod 3 <> 0) And n Mod 2 = 0) Then
        Print #2, "2의 배수"
    Else
        Print #2, "그 외의 숫자"
    End If
    Close #1, #2
End Sub

```

3) Select-Case문

하나의 조건에 대해서 3개 이상의 경우의 수가 있을 때 사용한다.

〈표Ⅱ.5.2〉 Select~Case문의 기본 구조

기본 구조	순 서 도
<pre> Selct Case 조건식 Case 경우1 문장1 Case 경우2 문장2 Case 경우3 문장3 Case Else 문장4 End Select 문장5 </pre>	<pre> graph TD 조건[조건] --> 경우1{경우 1} 경우1 -- 예 --> 문장1[문장 1] 경우1 -- 아니오 --> 경우2{경우 2} 경우2 -- 예 --> 문장2[문장 2] 경우2 -- 아니오 --> 경우3{경우 3} 경우3 -- 예 --> 문장3[문장 3] 경우3 -- 아니오 --> 문장4[문장 4] 문장4 --> 문장5[문장 5] </pre>

조건식의 결과에 따라 Case가 정해진다. 만약 조건식의 값이 경우2이면 문장2만을 수행하고 Select~Case문을 빠져나간다. 조건의 결과 값이 경우1, 2, 3에 해당되지 않으면 Case Else문 속의 문장4를 수행하게 된다. Select~Case문의 끝에는 End Select 문장이 있어야 한다.

Case문의 값의 형태는 다양하게 올 수 있다.

〈표Ⅱ.5.3〉 Case문의 형태

형태	설명	사용 예
Case 값1	하나의 값일 경우 수행	Case 100
Case 값1, 값2, 값3	여러 개의 값일 경우 수행	Case 10, 9
Case 시작값 To 최종값	특정한 범위 안의 값일 경우 수행	Case 10 to 100
Case Is 조건식	특정한 조건이 참일 경우 수행	Case Is >=80

Select~Case문의 예제로 점수를 입력받아 점수가 90~100점이면 “우수”, 70~89점이면 “보통”, 0~69점이면 “노력 요함”으로 나타내려고 한다.

```

Option Explicit
Sub main()
    Dim n As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n
    Select Case n \ 10
        Case 10, 9
            Print #2, "우수"
        Case 7, 8
            Print #2, "보통"
        Case Else
            Print #2, "노력요함"
    End Select
    Close #1, #2
End Sub

```



문제 II.5.2

두 숫자와 사칙연산 기호를 입력받아 계산하는 프로그램을 작성하시오.

입력 예) 50 2 *

출력 예) 50 * 2 = 100

<풀이>

```

Option Explicit
Sub main()
    Dim n As Integer, m As Integer
    Dim op As String
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n, m
    Input #1, op
    Select Case op
        Case "*"
            Print #2, n & op & m & " = " & (n * m)
        Case "/"
            Print #2, n & op & m & " = " & (n / m)
        Case "+"
            Print #2, n & op & m & " = " & (n + m)
        Case "-"
            Print #2, n & op & m & " = " & (n - m)
    End Select
End Sub

```

```

Case Else
Print #2, "잘못된 연산자 입력"
End Select
Close #1, #2
End Sub

```

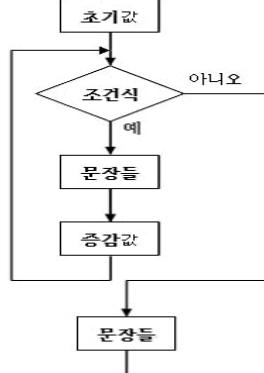
나. 반복문

프로그램의 일부분을 반복하여 수행하도록 하는 것을 반복문이라 한다. 반복문의 종류에는 For~Next문, Do While~Loop문, Do Until~Loop문, While~Wend문, For Each~Next문 등이 있다. 경우에 따라 If문과 Exit문을 이용하여 반복문을 빠져나가게 하기도 한다.

1) For~Next문

반복문 중 가장 많이 사용되는 제어문으로서, 초기값과 최종값 또는 반복 횟수를 정확하게 알고 있는 경우에 많이 사용된다. 형식은 다음과 같다.

〈표 II.5.4〉 For문의 기본 구조

기본 구조	순서도
For 변수=초기값 To 최종값 [Step 증감값] 반복할 문장들 [Exit For] Next 변수	 <pre> graph TD A[초기값] --> B{조건식} B -- 예 --> C[문장들] C --> D[증감값] D --> C B -- 아니오 --> E[문장들] </pre>
<ul style="list-style-type: none"> ■ 초기값 : 반복을 제어하는 제어변수의 초기값을 지정 ■ 조건식 : 반복문을 수행하기 위한 조건문, 조건이 참이면 문장들을 반복 수행하고, 거짓이면 For문을 빠져나감 ■ 증감값 : 제어변수의 증가 혹은 감소 등의 변화 ■ 초기값, 조건식, 증감값은 생략할 수도 있다. 	

예로 n부터 m까지의 합을 구하는 프로그램을 살펴보자.(단,m>n)

```

Option Explicit
Sub main()
    Dim n As Integer, m As Integer
    Dim sum As Integer, i As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n, m
    For i = n To m
        sum = sum + i
    Next i
    Print #2, n & "부터 " & m & "까지 합 = " & sum
    Close #1, #2
End Sub

```



문제 II.5.3

다음은 어떤 프로그램의 일부이다. 다음 프로그램을 실행시킨 후 C의 값은?

(2007 중등)

```

c = 0
For I = 1 To 10
    c = c + I
    I = I * 2
Next I

```

- ① 11 ② 12 ③ 13 ④ 14 ⑤ 15

<풀이> ⑤

For문은 기본적 제어변수를 1씩 증가시킨다. 따라서 $c=1+2+4+8$ 만 수행한다.



문제 II.5.4

10부터 n까지 3의 배수와 4의 배수가 아닌 정수들의 합을 구하시오. ($n > 10$)

<풀이>

```

Option Explicit
Sub main()
    Dim n As Integer
    Dim sum As Integer, i As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n

```

```

For i = 10 To n
    If (((i Mod 3) <> 0) And ((i Mod 4) <> 0)) Then
        sum = sum + i
    End If
    Next i
    Print #2, sum
    Close #1, #2
End Sub

```

2) Do ~ Loop문

Do~Loop문은 While문과 Until문 중 어느 것이 포함되어 있는지와 조건이 반복할 문장들보다 앞에 있는지 뒤에 있는지에 따라 크게 4종류로 나눠진다. While문이 포함된 Do~Loop문은 While 다음의 조건을 판단하여 **조건이 참이면 계속 반복할 문장을 실행하는 반복문**이다. 이에 반해 Until문이 포함된 반복문은 Until 뒤의 조건을 판단하여 조건이 거짓이면 계속 반복할 문장을 반복 수행하고 참이면 끝나는 반복문이다. 다음 표에서 2가지 종류를 살펴보자.

〈표II.5.5〉 Do ~ Loop문

종류별 기본 구조	설명
Do While 조건 반복할 문장들 [Exit Do] Loop	조건이 앞에 있어서 먼저 조건을 판단하여 참이면 계속 반복 수행한다. 처음부터 조건이 거짓이면 반복문 안에 있는 문장들은 한 번도 수행하지 않는다.
Do 반복할 문장들 [Exit Do] Loop While 조건	조건이 마지막에 있기 때문에 반복할 문장을 먼저 수행한 뒤에 조건을 판단하여 참이면 반복 수행한다. 조건을 나중에 판단하기 때문에 최소한 한번은 수행한다.

Do~Loop문의 2가지 경우에 대해 1부터 10까지의 합을 구하는 프로그램을 작성해 보자. 특히, 조건을 잘 살펴보자.

Do While ~ Loop문	Do ~ Loop While문
<pre>Option Explicit Sub main() Dim K, sum As Integer K = 1 sum = 0 Do While K <= 10 sum = sum + K K = K + 1 Loop Debug.Print sum End Sub</pre>	<pre>Option Explicit Sub main() Dim K, sum As Integer K = 1 sum = 0 Do sum = sum + K K = K + 1 Loop While K <= 10 Debug.Print sum End Sub</pre>



문제 II.2.5

input.txt 파일에서 입력자료 개수와 입력자료를 읽어와 최대값과 최소값의 차 이를 output.txt 파일로 출력하시오.(단, Do While문으로 작성할 것)

입력 예)

```
5
20 30 25 19 40
```

출력)

```
21
```

<풀이>

```
Option Explicit
Sub main()
    Dim n As Integer, data As Integer
    Dim max As Integer, min As Integer
    Dim i As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n
    Do While i < n
        Input #1, data
        If i = 0 Then
            min = data
            max = min
        End If
```

```

If max < data Then
    max = data
ElseIf (min > data) Then
    min = data
End If
i = i + 1
Loop
Print #2, max - min
Close #1, #2
End Sub

```

3) While ~ Wend문

While 뒤의 조건식을 평가하여 참이면 반복할 문장 실행, 거짓이면 빠져나간다. While~Wend문의 특징은 반복 중에 중지할 수가 없다는 점이 Do While~Loop문과 다르다. 즉, Exit문을 이용하여 While~Wend문을 빠져나갈 수 없다는 말이다.

4) 분기문 : Exit문

Exit문은 반복문이나 프로시저 등의 블록(영역)을 특정한 상황이 벌어지면 강제로 블록을 빠져나가게 할 때 사용한다. Exit문의 종류는 다음과 같다.

〈표 II.2.23〉 Exit문

종 류	설 명
Exit Do	Do ~ Loop문을 빠져나간다.
Exit For	For ~ Next문을 빠져나간다.
Exit Function	함수 프로시저를 종료한다.
Exit Sub	서브 프로시저를 종료한다.



연습문제



1부터 10까지 홀수의 합을 구하는 프로그램이 아닌 것은?(2005 중등)

① S = 0
For I = 1 To 5
 S = S + I
 Next I
③ I = 1: S = 0
 While I < 5
 S = S + I
 I = I + 1
 Wend
⑤ I = 0: S = 0
 Do
 I = I + 1
 S = S + I
 Loop While I < 5

② S = 0
For I = 0 To 4
 S = S + I + 1
 Next I
④ I = 0: S = 0
 While I < 5
 I = I + 1
 S = S + I
 Wend



10부터 1000까지 N으로 나눈 나머지를 중 홀수들만 합을 구하시오.



두 개의 양수를 입력받아 두 숫자 중 큰 수를 계속 합하려고 한다. 단, 두 숫자 중 하나라도 0이 입력되면 계산을 종료하고 결과를 출력하시오.



가로 N행, 세로 N열에 대해 아래와 같이 '#표시가 역삼각형의 형태로 출력되도록 만드시오.

입력) N=3 일 때

출력 예)

```
# # #
# #
#
```



10이상 40미만의 나이를 입력받아 10대, 20대, 30대 인지를 나타내는 프로그램을 작성하시오.

입력) 25

출력) 25는 20대



다음은 무엇을 구하는 프로그램인가?(2005 중등)

```
C = N - 1
For I = 2 To N
    For J = 2 To I - 1
        If (I Mod J = 0) Then
            C = C - 1
        Exit For
    End If
Next J, I
Debug.Print C
```

- ① N의 약수의 개수
- ② N이하인 소수의 개수
- ③ N이하인 홀수의 개수
- ④ N이하인 짝수의 개수
- ⑤ N이하인 합성수의 개수



문자 학점을 숫자 학점으로 바꾸는 프로그램을 작성하시오. 문자 학점은 A, B, C, D 및 F가 있다. 그리고 F를 제외한 문자 학점 다음에는 “+”가 나올 수 있다. 문자 학점 A, B, C, D 그리고 F에 대한 숫자 학점은 4, 3, 2, 1과 0이다. “+”는 숫자 학점을 0.5 증가시킨다. 예를 들어서 “B+”는 3.5가 된다. 실행 파일은 PR1.EXE라 한다. (2005 교원)

<조건 1> 입력 시, 문자 학점에 해당되는 A+, A, B+, B, C+, C, D+, D, F 이 외의 학점에 대해서는 “0(숫자)”를 출력한다. (단, 대소문자를 구분 하지 않는다. 예를 들어, “a”도 “A”와 같이 처리한다.)

<입력형식> 사용자로부터 문자 학점을 입력 받는다.

<출력형식> 출력 파일은 OUTPUT.TXT라 한다. 이 파일에는 문자 학점에 해당되는 숫자 학점이 출력된다. 예를 들어 사용자 입력에서 “B+”이 입력되면 그 결과는 “3.5”가 출력되어야 한다.

<입출력 예>

입력파일(INPUT.TXT)

B+

출력파일(OUTPUT.TXT)

3.5



<그림 1>과 같이 9×9 격자판에 쓰여진 81개의 자연수가 주어질 때, 이들 중 최대값을 찾고 그 최대값이 몇 행 몇 열에 위치한 수인지 구하는 프로그램을 작성하시오. (2007 고등)

예를 들어, 다음과 같이 81개의 수가 주어지면

	1열	2열	3열	4열	5열	6열	7열	8열	9열
1행	3	23	85	34	17	74	25	52	65
2행	10	7	39	42	88	52	14	72	63
3행	87	42	18	78	53	45	18	84	53
4행	34	28	64	85	12	16	75	36	55
5행	21	77	45	35	28	75	90	76	1
6행	25	87	65	15	28	11	37	28	74
7행	65	27	75	41	7	89	78	64	39
8행	47	47	70	45	23	65	3	41	44
9행	87	13	82	38	31	12	29	29	80

<그림 1>

이들 중 최대값은 90이고, 이 값은 5행 7열에 위치한다.

실행파일의 이름은 MAX.EXE로 하고, 프로그램의 실행시간은 1초를 넘을 수 없다. 부분 점수가 주어질 수 있다.

〈입력형식〉 입력 파일의 이름은 INPUT.TXT로 한다. 첫 째 줄부터 아홉 번째 줄까지 한 줄에 아홉 개씩 자연수가 주어진다. 주어지는 자연수는 100보다 작다.

〈출력형식〉 출력 파일의 이름은 OUTPUT.TXT로 한다. 첫째 줄에 최대값을 출력하고, 둘째 줄에 최대값이 위치한 행 번호와 열 번호를 빈칸을 사이에 두고 차례로 출력한다. 최대값이 두 개 이상인 경우 그 중 한 곳의 위치를 출력한다.

입력파일(INPUT.TXT)

3 23 85 34 17 74 25 52 65
10 7 39 42 88 52 14 72 63
87 42 18 78 53 45 18 84 53
34 28 64 85 12 16 75 36 55
21 77 45 35 28 75 90 76 1
25 87 65 15 28 11 37 28 74
65 27 75 41 7 89 78 64 39
47 47 70 45 23 65 3 41 44
87 13 82 38 31 12 29 29 80

출력파일(OUTPUT.TXT)

90
5 7



9개의 서로 다른 자연수가 주어질 때, 이들 중 최대값을 찾고 그 최대값이 몇 번째 수인지를 구하는 프로그램을 작성하시오. (2007 초등)

예를 들어, 서로 다른 9개의 자연수 3, 29, 38, 12, 57, 74, 40, 85, 61이 주어지면 이들 중 최대값은 85이고, 이 값은 8번째 수이다.

실행파일의 이름은 AA.EXE로 하고, 프로그램의 실행시간은 1초를 넘을 수 없다. 부분 점수가 주어질 수 있다.

〈입력형식〉 입력 파일의 이름은 INPUT.TXT로 한다. 첫 째 줄부터 아홉 번째 줄까지 한 줄에 하나의 자연수가 주어진다. 주어지는 자연 수는 100보다 작다.

〈출력형식〉 출력 파일의 이름은 OUTPUT.TXT로 한다. 첫째 줄에 최대 값을 출력하고, 둘째 줄에 최대값이 몇 번째 수인지를 출력한다.

〈입력과 출력의 예〉

입력파일(INPUT.TXT)

3
29
38
12
57
74
40
85
61

출력파일(OUTPUT.TXT)

85
8



정렬되지 않은 학생들의 임의의 점수를 입력하여 석차를 계산하는 프로그램을 작성하시오. 점수는 동점이 있을 수 있으며, 이러한 경우 같은 석차로 처리한다. 예를 들어 5명의 점수 100, 90, 76, 60, 90이 입력되었다면 석차는 2등이 2명이고 3등은 없다. (단, 점수가 가장 높은 학생을 1등으로 한다. 실행 파일은 “PR2.EXE”이다.)

〈입력형식〉 처리할 점수의 개수와 처리할 점수 데이터는 “INPUT2.TXT” 파일에서 입력받는다.

- 1) 입력 파일의 첫 번째 줄은 처리할 점수의 개수
- 2) 입력 파일의 두 번째 줄은 처리할 점수 데이터

(단, 각각의 점수는 빈칸으로 구별)

〈출력형식〉 석차를 계산한 후 점수와 석차를 “OUTPUT.TXT” 파일로 출력한다.

입력파일(INPUT.TXT)

10
60 34 55 100 90 76 60 90 80 87

출력파일(OUTPUT.TXT)

60	7
34	10
55	9
100	1
90	2
76	6
60	7
90	2
80	5
87	4



풀



①

②

```
Option Explicit
Sub main()
    Dim i As Integer, n As Integer
    Dim sum As Integer, temp As Long
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n
    For i = 10 To 20
        temp = (i Mod n)
        If temp Mod 2 <> 0 Then sum = sum + temp
    Next i
    Print #2, "result : " & sum
    Close #1, #2
End Sub
```

③

```
Option Explicit
Sub main()
    Dim n As Integer, m As Integer, sum As Long
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Do While (1)
        Input #1, n, m
        If (n = 0 Or m = 0) Then Exit Do
        If (n > m) Then
            sum = sum + n
        Else
            sum = sum + m
        End If
    Loop
    Print #2, sum
    Close #1, #2
End Sub
```

4

```
Option Explicit
Sub main()
    Dim i As Integer, j As Integer, n As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, n
    For i = n To 1 Step -1
        For j = n - 1 To 1 Step -1
            If j < i Then Exit For
            Print #2, " ";
            Next j
        For j = i To 1 Step -1
            Print #2, "# ";
            Next j
        Print #2, ""
    Next i
    Close #1, #2
End Sub
```

5

```
Option Explicit
Sub main()
    Dim age As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, age
    Select Case age \ 10
        Case 3
            Print #2, age & "는 30대"
        Case 2
            Print #2, age & "는 20대"
        Case 1
            Print #2, age & "는 10대"
        Case Else
            Print #2, "10대~30대가 아닙니다."
    End Select
    Close #1, #2
End Sub
```

6 ②



```
Option Explicit
Sub main()
    Dim data As String
    Dim result As Double
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, data
    data = UCASE(data)
    Select Case Left(data, 1)
        Case "A"
            result = 4
        Case "B"
            result = 3
        Case "C"
            result = 2
        Case "D"
            result = 1
    End Select
    If Right(data, 1) = "+" Then result = result + 0.5
    End If
    Print #1, result
    Close #1, #2
End Sub
```



```
Option Explicit
Sub main()
    Dim row As Integer, col As Integer, max As Long
    Dim i As Integer, j As Integer, data As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    For i = 1 To 9
        For j = 1 To 9
            Input #1, data
            If (data > max) Then
                max = data
                col = j
                row = i
            End If
        Next
    Next
    Print #2, max
    Print #2, " " & row & " " & col
    Close #1, #2
End Sub
```

9

```
Option Explicit
Sub Main()
    Dim max As Integer, index As Integer, a As Integer, i As Integer
    Open "input.txt" For Input As #1
    For i = 1 To 9
        Input #1, a
        If a > max Then
            max = a
            index = i
        End If
    Next
    Open "output.txt" For Output As #2
    Print #2, max
    Print #2, index
    Close #1, #2
End Sub
```

10

```
Option Explicit
Dim dice(2 To 12) As Long, i As Long
Dim first As Long, second As Long
Sub main()
    Open "output.txt" For Output As #1
    Randomize
    For i = 1 To 3600
        first = Int(Rnd() * 6 + 1): second = Int(Rnd() * 6 + 1)
        dice(first + second) = dice(first + second) + 1
    Next i
    For i = 2 To 12
        Print #1, CStr(i) + "--" + CStr(dice(i))
    Next i
    Close #1
End Sub
```

6. 프로시저

프로그래밍을 보다 쉽고 편리하게 관리하기 위해서 프로그램을 논리적으로 작은 단위로 나누어 프로그래밍을 하게 된다. 이런 프로그램의 논리적 작은 단위를 **프로시저(procedure)**라고 한다. Visual Basic에서 사용자가 정의하여 사용할 수 있는 프로시저의 종류에는 일반적으로 반환 값이 있는 경우에 사용하는 함수 프로시저와 반환 값이 없는 경우에 사용하는 서브 프로시저가 있다.

가. 함수(Function) 프로시저

매개 변수에 의해 정보를 받아서 내용을 처리하여 처리된 결과를 반환하는 경우에 많이 사용한다.

1) 함수(Function) 프로시저의 선언

● 형식

```
[ Private | Public | Static ] Function 프로시저 이름 (매개변수) [ As 자료형 ]
문장들
[Exit Function]
문장들
[함수명 = 식]
End Function
```

- ① 접근 지정자 : 프로시저를 호출할 수 있는 범위를 나타내는 것으로서, Private, Public, Static 중에서 사용된다. 생략한 경우에는 Public이 기본적으로 설정된다. Private은 선언된 프로시저는 호출 범위가 프로시저가 포함되어 있는 모듈이나 폼만 호출하여 사용할 수 있다. Public은 전역 프로시저로 만드는 것으로 선언된 프로시저는 다른 모듈이나 폼에서도 호출하여 사용이 가능하다. Static의 경우엔 ‘정적’, ‘고정’ 의미의 프로시저로서, Sub 프로시저 종료 후에도 프로시저 안의 지역 변수가 값을 기억하고 있다. 프로그램이 완전히 종료되면 사라진다.

- ② As 자료형 : 함수도 변수처럼 자신의 자료형을 가진다.
③ 매개변수 : 서로 다른 프로시저 간에 정보를 교환하기 위해 사용하는 연

결 변수로서, 일반적으로 호출하는 쪽의 매개 변수를 실인수라고 하고, 호출되어지는 서브 프로시저의 매개변수를 가인수라고 한다.

- ④ Exit Function : 조건이 참이면 함수 프로시저를 종료한다.
- ⑤ 함수명 = 식 : 함수명을 통해 반환하는 값이 있을 경우 사용한다.
- ⑥ End Function : 함수 프로시저를 종료한다.

2) 함수 프로시저 호출

함수를 호출할 때에는 반환되는 결과값이 있을 경우와 반환 결과값이 없을 경우로 나누어 볼 수 있다. 만약 후자의 경우에는 반환되는 값이 없으므로 변수에 대입시켜 줄 필요가 없다.

● 함수 호출문의 형식

변수 = 함수 프로시저 이름(매개 변수) '반환되는 결과값이 있을 때
또는

함수 프로시저 이름(매개 변수) '반환되는 결과값이 없을 때

다음 예제는 두 a, b를 입력받아 a와 b의 평균을 함수 프로시저에서 계산을 하고, 출력은 main() 서브 프로시저에서 하는 프로그램을 작성한 것이다.

```
Option Explicit
Sub main()
    Dim a As Integer, b As Integer
    Dim result As Single
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, a, b
    result = AV(a, b)                                  ' 함수 프로시저 호출- 괄호가 있음
    Print #2, result
    Close #1, #2
End Sub
```

' 호출되어지는 함수 프로시저

```
Function AV(x As Integer, y As Integer) As Single
    AV = CSng((x + y) / 2)                                  ' Csng()는 Single형으로 변환
End Function
```

나. 서브 프로시저

단순히 공통적으로 반복되는 작업을 하나의 독립적인 논리적 단위로 만들어 프로그래밍하려고 할 때 사용하는 프로시저로 반환되는 값은 없으며, 인수를 통해 필요한 데이터를 주고받는다.

1) 서브 프로시저의 선언

서브 프로시저의 선언 형식은 다음과 같다.

● 선언 형식

```
[ Private | Public | Static ] Sub 프로시저 이름 (매개변수)
    변수 선언들
    문장들
    [Exit Sub]
    문장들
End Sub
```

2) 서브(Sub) 프로시저 호출

서브 프로시저 호출문의 형식은 다음과 같다.

● 형식

프로시저 이름 인수1, 인수2, . . . , 인수n

다음 예제는 두 a, b를 입력받아 a와 b의 평균을 함수 프로시저에서 계산을 하고, 출력은 main() 서브 프로시저에서 하는 프로그램을 작성한 것이다.

```
Option Explicit
Sub main()
    Dim a As Integer, b As Integer
    Dim result As Single
    Open "input.txt" For Input As #1
    Input #1, a, b
    AV a, b          ' 함수 프로시저 호출- 괄호가 있음
End Sub
Sub AV(x As Integer, y As Integer)   ' 호출되어지는 함수 프로시저
    Open "output.txt" For Output As #2
    Print #2, CSng((x + y) / 2)        ' CSng()는 Single형으로 변환
End Sub
```

다. 내장 함수

내장(Library, 라이브러리) 함수는 사용자들이 많이 사용하는 함수들을 미리 작성해서 컴퓨터의 내부에 저장시켜 놓고 사용자가 필요할 때 바로 사용할 수 있는 함수를 말한다. 내장함수에는 수학함수, 문자열 처리 함수, 날짜·시간 함수 등이 있다. 자주 사용되는 함수들에 대해 살펴보도록 하자.

1) 수학 함수

수학과 관련된 다양한 함수들을 제공하고 있다. 다음 <표 II.6.1>에서 살펴보자.

<표 II.6.1> 수학 관련 함수들

함수 종류	설명	사용 예	결과값
Int(n)	실수 n을 정수로 변환하는 함수로서, n보다 크지 않는 최대 정수값을 반환	Int(-10.2345)	-11
Fix(n)	실수 n을 정수로 변환하는 함수로서 n의 소수점을 모두 없앤 정수값을 반환	Fix(-10.2345)	-10
Abs(n)	n의 절대값을 반환	Abs(-10)	10
Sqr(n)	n의 제곱근을 반환	Sqr(9)	3

함수 종류	설명	사용 예	결과값
Rnd()	0부터 1 사이의 난수값(무작위 추출값)을 반환한다. 보통, Randomize문으로 먼저 난수 발생기를 초기화해야만 계속해서 다른 난수가 발생한다.	Rnd()	0.7055475

2) 문자열 처리 함수

문자열과 관련된 함수들이다. 많이 사용되므로 주의 깊게 살펴보도록 하자.

〈표Ⅱ.6.2〉 문자열 관련 함수들

종 류	설 명	사용 예	결과값
Len(s)	문자열의 길이를 반환	ln = Len("ABCD")	4
Str(n)	숫자를 문자열로 변환하여 반환	st = Str(1234)	1234 (문자)
Val(s)	문자열 내의 숫자문자를 수치로 변환하여 반환	n = Val("1234")	1234 (숫자)
Asc(s)	문자열의 첫 글자에 대응하는 아스키 코드값을 반환	n = Asc("ABCD")	65
Chr(n)	지정된 아스키(ASCII) 코드값에 대응하는 문자를 반환	ch = Chr(65)	A
LCase(s)	문자열에서 대문자를 소문자로 변환하여 반환	LCase("ABcd")	abcd
UCase(s)	문자열에서 소문자를 대문자로 변환하여 반환	UCase("ABcd")	ABCD
Left(s)	문자열에서 왼쪽부터 지정된 개수만큼 반환	Left("ABCDEF",2)	AB
Right(s)	문자열에서 오른쪽부터 지정된 개수만큼 반환	Right("ABCDEF",2)	EF
Mid(s)	문자열에서 지정된 위치에서 정해진 개수만큼 반환	Mid("ABCDEF", 2, 3)	BCD
Trim(s)	문자열의 왼쪽과 오른쪽의 공백을 제거하여 반환	Trim(" ABCD ")	ABCD

라. 변수의 범위 (지역변수와 전역변수)

변수는 사용할 수 있는 유효 범위에 따라 전역변수와 지역변수로 나눈다.

1) 지역변수

지역변수는 함수의 내부에서 선언되어 함수 내에서만 사용되는 변수이며, 내부변수라고도 한다. 변수는 변수가 선언될 때 메모리 공간이 부여되고, 함수가 끝나면 메모리 공간을 운영체제에게 회수 당해 없어진다.

2) 전역변수

전역변수는 함수 외부에 선언되어 선언된 이후의 모든 함수에서 사용이 가능한 변수를 말하며, 외부변수라고도 한다. 프로그램이 실행이 시작될 때 선언이 되어 끝날 때까지 계속 메모리 공간이 남아 있다. 전역변수와 지역변수가 이름이 같이 존재하면 지역변수가 우선이다.

```

Option Explicit
Dim K As Integer           ' 전역변수
Sub main()
    Dim a As Integer, b As Integer      ' main()함수의 지역변수 선언
    Open "output2.txt" For Output As #2
    a = 10
    b = 20
    abc
    Print #2, "a = " & a & "   K =" & K
End Sub                      ' 파일 닫는 문장 생략 가능
Sub abc()
    Dim a As Integer
    a = 5                     ' sum()함수의 지역변수 선언
    K = 50                    ' K는 전역변수
    K = K + a
    Print #2, "a = " & a & "   K =" & K
End Sub

```

출 력	a = 5 K =55
결 과	a = 10 K =55

마. 되부름 함수

자기 자신을 부르는 함수를 되부름 함수(recursive function) 또는 재귀적 호출 함수라고 하고, 보통 동일한 기능의 반복을 처리하기 위해 사용한다. 1 1 2 3 5 8 과 같은 피보나치 수열은 보통 되부름 함수로 표현을 많이 한다. 다음 예의 결과는 얼마인지 계산해 보자.(2002 경남)

```
Function f(n, k)
    If ((k = 1) Or (n = k)) Then
        f = 1
    Else
        f = k * f(n - 1, k) + f(n - 1, k - 1)
    End If
End Function
Sub main()
    Debug.Print f(5, 3)
End Sub
```

$f(5,3)$ 을 호출하면 $f()$ 함수의 f 는 처음에 $3*f(4,3)+f(4,2)$ 이 되고. 다시 $f(4,3)$ 과 $f(4,2)$ 는 각각의 $f()$ 함수를 호출하게 된다. 결국 $f(1,0)$ 이 되면 1을 되돌려준다. 따라서 출력은 25가 된다.



문제 I 6.1

다음 함수에서 recursive(5)를 호출할 때 출력되는 “*”의 개수는? (2006교원)

```
Private Sub recursive(n As Integer)
    If n > 1 Then
        recursive n / 2
        recursive n / 2
    End If
    Debug.Print "*"
End Sub
```

- ① 4 ② 5 ③ 6 ④ 7 ⑤ 8

<풀이> ④



연습문제



다음 프로그램을 실행시키면 직접 실행창에 어떤 결과가 출력되는가?

(2002 초등)

```
Sub Main()
    Debug.Print F(7)
End Sub
Function F(N)
    If N <= 2 Then
        F = N
    Else
        F = F(N - 1) * F(N - 2)
    End If
End Function
```

- ① 64 ② 128 ③ 196 ④ 256 ⑤ 384



다음 함수는 무엇을 하는 함수인가? (2004 고등)

```
Function F(A, B) As Integer
    F = ((A + B) + Abs(A - B)) / 2
End Function
```

- ① 두 값 중 큰 값을 구하는 함수 ② 두 값 중 작은 값을 구하는 함수
 ③ 두 값의 평균값을 구하는 함수 ④ 두 값의 차를 구하는 함수
 ⑤ 한 값을 다른 한 값으로 나눈 나머지를 구하는 함수



다음 프로그램을 실행시켰을 때의 출력 결과는?(2007 교원)

```
Private Sub Form_Load()
    Dim a As Integer, b As Integer, c As Integer
    a = 30
    b = 40
```

```

If a > b Then
    c = gcd(a, b)
Else
    c = gcd(b, a)
End If
Debug.Print c
End Sub
Private Function gcd(m As Integer, n As Integer) As Integer
    If n = 0 Then
        gcd = m
    Else
        gcd = gcd(n, m Mod n)
    End If
End Function

```

- ① 10 ② 15 ③ 20 ④ 25 ⑤ 30



다음 프로그램을 실행시켰을 때의 출력 결과는?(2007 교원)

```

Private Sub Form_Load()
    Dim i As Integer, j As Integer
    j = 0
    For i = 0 To 5 Step 1
        If (isFunction(4 * i + 3)) Then
            j = j + (4 * i + 3)
        End If
    Next
    Debug.Print j
End Sub
Function isFunction(n As Integer) As Boolean
    For i = 2 To n - 1 Step 1
        If (n Mod i = 0) Then
            isFunction = False
            Exit Function
        End If
    Next
    isFunction = True
End Function

```

- ① 63 ② 40 ③ 78 ④ 55 ⑤ 50



다음 실행 프로그램을 실행시켰을 때의 실행 결과는? (2007 교원)

```

Private Sub Form_Load()
    Dim a(5) As Integer
    Dim b As Integer
    Dim i As Integer
    b = 0
    For i = 0 To 5 Step 1
        setNum a, 6, i
    Next
    For i = 0 To 5 Step 1
        b = b + a(i)
    Next
    Debug.Print b
End Sub

Private Sub setNum(a() As Integer, m As Integer, n As Integer)
    Dim i As Integer
    Dim f As Boolean
    Randomize
    If n = 0 Then
        a(n) = Int(Rnd() * 6 + 1)
    Else
        Do While (1)
            a(n) = Int(Rnd() * 6 + 1)
            f = True
            For i = 0 To n - 1 Step 1
                If a(i) = a(n) Then
                    f = False
                    Exit For
                End If
            Next
            If f = True Then
                Exit Do
            End If
        Loop
    End If
End Sub

```

- ① 20 ② 21 ③ 22 ④ 23 ⑤ 24

⑥

다음 실행 프로그램을 실행시켰을 때의 실행 결과는? (2007 교원)

```
Private Sub Form_Load()
    Dim a As Integer
    a = recursive(10)
    Debug.Print a
End Sub

Private Function recursive(n As Integer) As Integer
    If n <= 1 Then
        recursive = n
    Else
        recursive = recursive(n - 2) + recursive(n - 1)
    End If
End Function
```

- ① 21 ② 22 ③ 33 ④ 34 ⑤ 55

⑦

한계값 N까지 친화수(완전수)의 쌍을 표현하고 쌍의 개수도 알려주는 프로그램을 작성하시오. 친화수가 서로 자신을 제외한 약수들의 합의 되는 수를 말한다. 예를 들면, 220은 약수의 합이 284이고 284의 약수의 합은 220으로 200과 284는 친화수이다. (단, N은 2000이하의 수)

입력(INPUT.TXT)

1500

출력(OUTPUT.TXT)

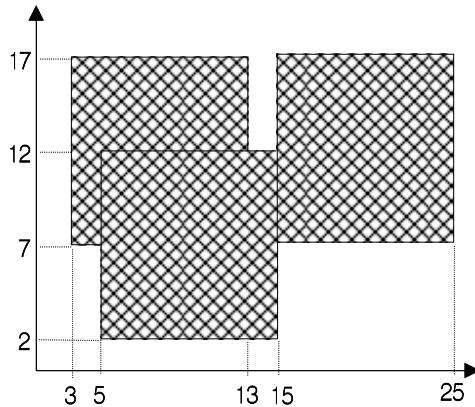
220, 284
284, 220
1184, 1210
1210, 1184
Total Count : 4

⑧

가로, 세로의 크기가 각각 100인 정사각형 모양의 흰색 도화지가 있다. 이 도화지 위에 가로, 세로의 크기가 각각 10인 정사각형 모양의 검은색 색종이를 색종이의 변과 도화지의 변이 평행하도록 붙인다. 이러한 방식으로

색종이를 한 장 또는 여러 장 붙인 후 색종이가 붙은 겸은 영역의 넓이를 구하는 프로그램을 작성하시오.(2007 초등)

예를 들어 흰색 도화지 위에 세 장의 겸은색 색종이를 아래 그림과 같은 모양으로 붙였다면 겸은색 영역의 넓이는 260이 된다.



실행파일의 이름은 BB.EXE로 하고, 프로그램의 실행시간은 1초를 넘을 수 없다. 부분 점수는 없다.

〈입력 형식〉 첫째 줄에 색종이의 수가 주어진다. 이어 둘째 줄부터 한 줄에 하나씩 색종이를 붙인 위치가 주어진다. 색종이를 붙인 위치는 두 개의 자연수로 주어지는데 첫 번째 자연수는 색종이의 왼쪽 변과 도화지의 왼쪽 변 사이의 거리, 두 번째 자연수는 색종이의 아래쪽 변과 도화지의 아래쪽 변 사이의 거리이다. 색종이의 수는 100이하이며, 색종이가 도화지 밖으로 나가는 경우는 없다.

〈출력 형식〉 첫째 줄에 색종이가 붙은 겸은 영역의 넓이를 출력한다.

〈입력과 출력의 예〉

입력(INPUT.TXT)

3
3 7
15 7
5 2

출력(OUTPUT.TXT)

260



풀

이

1 ④

2 ① Abs()함수는 절대값을 반환해 주는 프로시저임.

3 ①

4 ①

5 ②

6 ⑤

7

```

Option Explicit
Public Function fd(x As Integer) As Integer
    Dim i As Integer, sum As Integer, temp As Integer
    sum = 0
    For i = 1 To (x - 1)
        If (x Mod i = 0) Then sum = sum + i
    Next i
    fd = sum
End Function
Sub main()
    Dim i As Integer, sum As Integer
    Dim max As Integer, b As Integer, count As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, max
    For i = 2 To max
        sum = fd(i)
        If (i = fd(sum) And i <> sum And sum <= max) Then
            Print #2, i; sum
            count = count + 1
        End If
    Next i
    Print #2, "Total Count : " & count
End Sub

```



```

Option Explicit
Sub main()
    Dim n As Integer, x As Integer, y As Integer
    Dim i As Integer, j As Integer
    Dim area As Integer
    Dim p(100, 100) As Boolean
    Open "input.txt" For Input As #1
    Input #1, n
    Do While n > 0
        n = n - 1
        Input #1, x, y
        For i = x To x + 9
            For j = y To y + 9
                p(i, j) = True
            Next
        Next
        Loop
        Open "output.txt" For Output As #2
        For i = 1 To 99
            For j = 1 To 99
                If (p(i, j) = True) Then area = area + 1
            Next
        Next
        Print #2, area
        Close #1, #2
    End Sub

```

7. 배열

자료형이 같은 변수들을 하나로 묶어 관리하는 것을 배열이라고 하며 실제로 메모리에 저장될 때는 연속해서 저장된다. 배열을 선언할 때는 Dim문을 이용하고, 자료형과 몇 개의 변수가 필요한지를 적어준다. 실제 사용할 때에는 배열명에 첨자를 0부터 크기까지 사용이 된다. 배열은 선언할 때 배열의 크기를 설정하여 배열의 크기나 범위를 정하는 정적배열과 선언할 때 배열의 크기를 설정하지 않아 배열의 크기를 유동적으로 사용할 수 있는 동적배열이 있다.

가. 1차원 배열

1차원 배열은 첨자가 하나인 것으로 배열의 선언 형식은 다음과 같다.

● 배열 선언 형식

Dim 배열이름(상한값) As 자료형

아래와 같이 jumsu(4)가 선언되었다고 할 때, 메모리 공간은 다음과 같이 연속해서 배치된다.

배열 변수 선언 : Dim jumsu(4) As Integer

jumsu(0)	jumsu(1)	jumsu(2)	jumsu(3)	jumsu(4)	배열 요소
					메모리 저장공간

2byte 2byte 2byte 2byte 2byte 크기

1차원 배열 변수 선언 방식에 대해 다시 살펴보자.

● 1차원 배열 선언 형식

Dim 배열명(상한값) As 자료형

Dim 배열명(시작값 To 상한값) As 자료형

예를 들어 Dim K(3) As Integer로 배열 선언이 되면 배열 요소는 4개이며, 첨자는 0부터 3까지 사용된다.

다음 예제는 5명의 컴퓨터 점수를 입력받아 첫 줄에는 평균과 총점을 먼저 출력하고, 둘째 줄에는 5명의 점수를 출력하는 프로그램을 작성한 것이다.

```

Option Explicit
Sub main()
    Dim jumsu(4) As Integer
    Dim i As Integer, sum As Integer
    sum = 0
    For i = 0 To 4
        jumsu(i) = Val(InputBox("점수 입력"))
        sum = sum + jumsu(i)
    Next i
    Debug.Print " 평균 = " & sum / 5, " 합계 =" & sum
    Debug.Print " 입력자료 : ";
    For i = 0 To 4
        Debug.Print jumsu(i);
    Next i
End Sub

```

실행 결과	평균 = 84.4 합계 = 422 입력자료 : 85 75 68 94 100
-------	---



문제Ⅱ.7.1

1~100사이의 10개의 점수를 난수를 발생하여 배열에 저장한 뒤, 최고 점수와 입력 자료를 출력하는 프로그램을 작성하시오. (점수는 모두 정수이다)

<풀이>

```

Option Explicit
Sub main()
    Dim jumsu(1 To 10) As Integer
    Dim i As Integer, max As Integer
    max = 0
    Randomize
    For i = 1 To 10
        jumsu(i) = Int(Rnd() * 100 + 1)
        If max < jumsu(i) Then max = jumsu(i)
    Next i
    Debug.Print " 최대값 = " & max
    Debug.Print " 입력자료 : ";
    For i = 1 To 10
        Debug.Print jumsu(i);
    Next i
End Sub

```

실행 결과	최대값 = 96 입력자료 : 92 11 27 6 53 56 31 96 1 84
-------	--

나. 다차원 배열

배열의 첨자가 2개 이상 사용되었을 때를 다차원 배열이라고 한다. 첨자의 개수에 따라서 2차원 배열, 3차원 배열, 4차원 배열 등이 된다.

2차원 배열의 선언 형식은 다음과 같다.

● 2차원 배열 선언 형식

Dim 배열명(행, 열) As 자료형

Dim 배열명(행의 시작값 To 상한값, 열의 시작값 To 상한값)

Dim K(2, 3) As Long 으로 선언이 되었을 때의 배열 구성을 표로 그려보면 다음과 같다.

K(0,0)	K(0,1)	K(0,2)	K(0,3)
K(1,0)	K(1,1)	K(1,2)	K(1,3)
K(2,0)	K(2,1)	K(2,2)	K(2,3)

다음은 파일로부터 번호와 컴퓨터 점수를 입력받아 점수가 가장 높은 학생의 번호와 점수를 출력하는 프로그램이다.(단, 학생수는 1000명 이하, 동점은 없고, 번호가 0이면 계산을 마친다.)

```

Option Explicit
Sub main()
    Dim data(100, 2) As Integer, i As Integer
    Dim num As Integer, max As Integer
    num = 1
    max = 0
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    For i = 0 To 1000
        Input #1, data(i, 0), data(i, 1)
        If data(i, 0) = 0 Then Exit For
        If data(i, 1) > max Then
            max = data(i, 1)
            num = data(i, 0)
        End If
    Next
    Print #2, "번호 = " & num & " 점수 = " & max
End Sub

```



연습문제



다음은 어떤 프로그램의 일부이다. 배열 A가 아래와 같을 때 다음 프로그램을 실행시킨 후 J값으로 알맞은 것은?(2007 초등)

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)	A(10)
15	7	3	20	24	6	12	30	9	15

```

N = 10: I = 1: J = N + 1
Do
    Do
        I = I + 1
    Loop While A(I) <= A(1)
    Do
        J = J - 1
    Loop While A(J) >= A(1)
    If I < J Then Temp = A(I): A(I) = A(J): A(J) = Temp
Loop While I < J

```

- ① 5 ② 6 ③ 7 ④ 8 ⑤ 9



하나의 문장(2자 이상)를 입력받아 그 단어의 철자를 반대로 나열하는 프로그램을 작성하시오.

조건 1. 문자열 배열에 입력한 다음 이 배열을 출력하시오.

조건 2. 출력 시 공백은 없앤다.

입력 예) Oh! My god!

출력 예) !dogyM!hO



A(1)부터 A(10)에는 1부터 10까지의 서로 다른 수가 임의의 순서로 들어 있다. 다음 프로그램은 어떤 프로그램의 일부이다. 이 부분을 실행시켰을 때 k가 될 수 있는 값 중 최소값은? (2006 중등)

```

M = A(1); N = A(2)
For I = 3 To 10 Step 2
    J = I + 1
    If A(I) > M Then M = A(I)
    If A(J) < N Then N = A(J)
Next I
K = M - N

```

- ① -5 ② -1 ③ 1 ④ 5 ⑤ 9



다음 프로그램을 실행시켰을 때 출력되는 수는 몇 개인가?(2005 교원)

```

For I = 1 To 10
    W(I) = 0
Next I
For I = 1 To 10
    J = I
    While J <= 10
        W(J) = 1 - W(J)
        J = J + I
    Wend
Next I
For I = 1 To 10
    If W(I) = 1 Then Debug.Print I
Next I

```

- ① 1개 ② 2개 ③ 3개 ④ 4개 ⑤ 5개



다음 프로그램의 실행 결과는? (교원 2007)

```

Private Sub Form_Load()
    Dim t As Integer, n As Integer, m As Integer, i As Integer
    Dim s As Integer, c As Integer
    t = 0: n = 9: s = 3: c = 0
    Dim l(9) As Integer
    For i = 0 To 9 Step 1
        l(i) = i + 1
    Next
    Do While (t <= n)
        m = Fix((t + n) / 2)
    
```

```

If (l(m) < s) Then
    t = m + 1: c = c + 1
ElseIf (l(m) > s) Then
    n = m - 1: c = c + 1
Else
    Debug.Print CStr(c + 1) + "-" + CStr(m)
    Exit Do
End If
Loop
End Sub

```

- ① 2-2 ② 2-3 ③ 3-2 ④ 3-3 ⑤ 4-3



1부터 10 사이의 숫자가 적혀진 다섯 장의 카드가 주어진다. 그 중 세 장의 카드를 골라 그 합의 일의 자리수를 가장 크게 만들려고 한다. 예를 들어 다섯 장의 카드가 (7, 5, 5, 4, 9)인 경우 (7, 4, 9)를 선택하면 합이 20이 되어 일의 자리수는 0이 되고, (5, 5, 9)를 선택하면 합이 19가 되어 일의 자리수는 9가 된다. 세 장의 카드를 뽑아 그 합의 일의 자리수를 9보다 크게 만들 수는 없으므로 구하고자 하는 답은 9가 된다. 다섯 장의 카드 번호가 담긴 배열 $a[1] \sim a[5]$ 가 주어질 때, 이와 같이 구한 일의 자리수의 최대값 m 을 출력하는 프로그램을 작성하시오. (2006 중등)

입력파일(INPUT.TXT)

7 5 5 4 9

출력파일(OUTPUT.TXT)

9



1과 자신의 수로 나눠지는 수를 솟수(prime number)라고 한다. 예를 들면, 2, 3, 5, 7은 솟수지만 4, 6, 8, 9는 그렇지 않다. 1부터 100사이의 솟수를 출력하는 프로그램을 작성하시오. 실행파일은 PR1.EXE라 한다. (2004 교원)

〈출력 형식〉 출력파일은 OUTPUT.TXT라 한다. 각각의 솟수는 콤마로 구별한다. 한 줄에 솟수 10개씩 출력되도록 한다.

출력 예(OUTPUT.TXT)

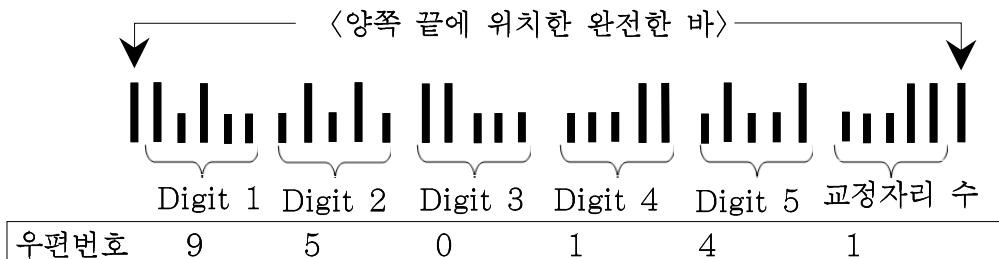
2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
31, 37, 41,

8

다량의 우편물을 보낼 때 우체국은 다량의 편지들을 빠르게 정렬하기 위하여 우편번호를 나타내는 바코드(Bar Code)의 사용을 장려한다. 이 우편번호 바코드는 바코드화된 다섯 자리 숫자 다음에 교정 자리수에 대한 인코딩이 위치하고, 양쪽 끝에 완전한 바가 추가된다. 교정 자리수는 다음과 같이 계산한다.

교정 자리수는 우편번호의 모든 자리수를 더한 합을 10의 배수로 만드는 수이다. 예를 들어, 우편 번호 95014는 자리수들의 합이 19이므로 이 합을 20으로 만들기 위한 교정 자리수는 1이다.

예)



우편 번호의 각 자리수와 교정 자리수는 아래 표에 의하여 인코딩 된다.

	7	4	2	1	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

표에서, 0은 높이가 1/2인 바를 나타내고 1은 완전한 바를 나타낸다. 자리수의 인코딩은 2개의 완전한 바와 3개의 높이가 1/2인 바의 조합이다. 바코드에 대응하는 자리수는 열의 가중치 7, 4, 2, 1, 0을 이용하여 쉽게 계산할 수 있다.

예를 들어, 01100은 $0 \times 7 + 1 \times 4 + 1 \times 2 + 0 \times 1 + 0 \times 0 = 60$ 된다. (단, 0은 예외이다)

우편번호를 입력하여, 바코드를 인쇄하는 프로그램을 작성하시오. 높이가 1/2인 바는 :로 나타내고 완전한 바는 |로 나타낸다. 실행파일은 PR2.EXE라 한다. (2005 교원)

〈입력 형식〉 사용자로부터 우편번호에 해당되는 숫자(5자리)를 입력 받는다.
입력에는 전혀 오류가 없다고 가정한다.

〈출력 형식〉 출력파일은 OUTPUT.TXT라 한다. 사용자가 입력한 숫자(5자리)에 해당되는 결과 값이 출력되어야 한다. 입력 예(95014)의 경우 코드로 변환된 ||:|:::||:||::::::||:|::|:::|||이 출력되어야 한다.

〈입출력 예〉

입력파일(INPUT.TXT)
95014

출력파일(OUTPUT.TXT)

||:|:::||:||::::::||:|::|:::|||



두 개의 주사위를 던져 주사위 윗면의 합을 계산하는 프로그램을 작성하되, 다음 그림과 같이 36개의 조합을 고려하고, 1차원 배열을 이용하여 각각의 주사위의 합이 나오는 횟수를 계산하시오.(단, 주사위는 난수발생기를 이용하며, 두 주사위를 3,600번 던진다고 가정하고, 실행파일은 pr1.exe라 한다) (2004 교원)

2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10
6	7	8	9	10	11
7	8	9	10	11	12

〈출력형식〉

출력 파일은 OUTPUT.TXT라 한다. 첫번째 열에는 각각의 주사위의 합을, 두번째 열에는 해당 주사위의 합이 나오도록 횟수를 출력한다. 출력형식은 오른쪽 표와 같다.

2-102
3-203
4-302
5-397
6-499
7-601
8-497
9-399
10-301
11-202
12-97



① ②

②

```
Option Explicit
Sub main()
    Dim i As Integer, le As Integer
    Dim data As String, temp(1000) As String
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    Input #1, data
    le = Len(data)
    For i = 0 To le - 1
        temp(i) = Mid(data, le - i, 1)
    Next
    For i = 0 To le
        If (temp(i) <> " ") Then Print #2, temp(i);
    Next
End Sub
```

③

④

⑤

6

```

Option Explicit
Sub main()
    Dim i As Integer, j As Integer, p As Integer, m As Integer, s As Integer
    Dim a(6) As Integer
    Open "input.txt" For Input As #1
    Open "output.txt" For Output As #2
    For i = 1 To 5
        Input #1, a(i)
    Next
    For i = 1 To 5
        s = s + a(i)
    Next
    For i = 1 To 3
        For j = i + 1 To 5
            p = (s - a(i) - a(j)) Mod 10
            If p > m Then m = p
        Next
    Next
    Print #2, m
    Close #1, #2
End Sub

```

7

```

Option Explicit
Sub main()
    Dim i As Integer, count As Integer
    Dim sosu(100) As Integer
    Open "output.txt" For Output As #1
    For i = 2 To 100
        If Prime(i) = True Then
            count = count + 1
            sosu(count) = i
        End If
    Next
    For i = 1 To count
        If i = count Then
            Print #1, "" & sosu(i);
        Else
            Print #1, sosu(i) & ", ";
        End If
        If i Mod 10 = 0 Then Print #1,
    Next
End Sub

```

```

Private Function Prime(n As Integer) As Boolean '소수인지 아닌지를 판단하는 함수
    Dim i As Integer
    Prime = True
    For i = 2 To Int(Sqr(n))
        If (n Mod i) = 0 Then
            Prime = False
            Exit For
        End If
    Next

```



Option Explicit

```

Dim data As Long, bar(6) As Long, lain(10, 5) As Long
Sub main()
    lain(1, 1) = 0: lain(1, 2) = 0: lain(1, 3) = 0
                lain(1, 4) = 1: lain(1, 5) = 1
    lain(2, 1) = 0: lain(2, 2) = 0: lain(2, 3) = 1
                lain(2, 4) = 0: lain(2, 5) = 1
    lain(3, 1) = 0: lain(3, 2) = 0: lain(3, 3) = 1
                lain(3, 4) = 1: lain(3, 5) = 0
    lain(4, 1) = 0: lain(4, 2) = 1: lain(4, 3) = 0
                lain(4, 4) = 0: lain(4, 5) = 1
    lain(5, 1) = 0: lain(5, 2) = 1: lain(5, 3) = 0
                lain(5, 4) = 1: lain(5, 5) = 0
    lain(6, 1) = 0: lain(6, 2) = 1: lain(6, 3) = 1
                lain(6, 4) = 0: lain(6, 5) = 0
    lain(7, 1) = 1: lain(7, 2) = 0: lain(7, 3) = 0
                lain(7, 4) = 0: lain(7, 5) = 1
    lain(8, 1) = 1: lain(8, 2) = 0: lain(8, 3) = 0
                lain(8, 4) = 1: lain(8, 5) = 0
    lain(9, 1) = 2: lain(9, 2) = 0: lain(9, 3) = 1
                lain(9, 4) = 0: lain(9, 5) = 0
    lain(0, 1) = 1: lain(0, 2) = 1: lain(0, 3) = 0
                lain(0, 4) = 0: lain(0, 5) = 0
    Open "input.txt" For Input As #1
    Input #1, data
    Close #1
    process
    output
End Sub

```

```

Sub process()
    Dim i As Long, hap As Long
    For i = 1 To 5
        bar(6 - i) = (data Mod 10)
        data = data \ (10)
        hap = hap + bar(6 - i)
    Next
    bar(6) = (hap \ 10 + 1) * 10 - hap
    Debug.Print bar(6)
End Sub
Sub output()
    Dim i As Long, j As Long
    Open "output.txt" For Output As #2
    Print #2, "|";
    For i = 1 To 6
        For j = 1 To 5
            If lain(bar(i), j) = 0 Then
                Print #2, ":";
            Else
                Print #2, "|";
            End If
        Next
    Next
    Print #2, "|"
    Close #2
End Sub

```



```

Option Explicit
Dim dice(2 to 12) as long, i as long
Dim first as long, second as long
sub main()
    open "output.txt" for output as #1
    randomize
    for i = 1 to 3600
        first = int(rnd() * 6 + 1) : second = int(rnd() * 6 + 1)
        dice(first + second) = dice(first + second) + 1
    next i
    for i = 2 to 12
        print #1, cstr(i) + "--" + cstr(dice(i))
    next i
    close #1
end sub

```

