2 변수

변수는 변할 수 있는 수를 담는 그릇이다.

원리를 알면 IT가 맛있다
IT COOKBOOK for Beginner
한빛 교재 시리즈

이 장에서 다룰 내용

1. 변수의 기본
2. 구조체
3. namespace
4. 형 변환

01_변수의 기본

❖ 변수의 정의

- 그릇: 값을 담을 수 있는 공간
- 메모리: 물리적으로 서로 구별이 가능한 식별자를 통해 접근할 수 있는 공간

❖ 변수의 구성요소

- 변수명(Variable Name)
- 변수형(Variable Type)

❖ 변수의 기본 형태와 사용 예

```
변수형 변수명1, 변수명2..
```

[기본 형태]

```
int var1, var2;
```

[사용 예]

❖ 변수명을 정하는 규칙

- 영문자, 숫자, _(underscore)를 혼용하여 만들 수 있다.
- _ 또는 알파벳 영문자로 시작해야 한다. 즉, 숫자로 시작할 수 없다.
- 알파벳 영문자의 대문자와 소문자를 구분한다. 즉, 변수 ABC와 변수 abc는 다른 변수다.
- C++에서 사용하는 예약어를 사용할 수 없다.
- 연결된 문자로만 만들어야 한다. 즉, 중간에 공백이 있을 수 없다.
- 특수문자를 사용할 수 없다.

변수명	사용 예	적용	이유
A	int A;	○	사용에 아무런 문제가 없다.
You	char You;	○	사용에 아무런 문제가 없다.
char	char char;	X	키워드와 이름이 동일하다.
public	int public;	×	public이라는 접근제한자와 이름이 동일하다.
1var	int 1var;	×	숫자로 시작할 수 없다.

01_변수의 기본

❖ 변수형

- 사용 컴파일러마다 조금씩 차이를 보이지만, 일반적으로 아래와 같은 기본적인 변수형을 지원한다
- 기본 변수형 앞에 signed / unsigned 키워드를 더하여 부호/미부호 변수를 생성할 수 있다.

변수형	사용 예	내용
bool	bool flag;	true, false
char	char str;	하나의 문자 데이터
short, int, long, long long	long temp;	부호가 있는 정수형
float, double	double bigNumber;	실수
void	void anything;	형이 없음

[기본예제 2-1] signed와 unsigned의 비교

```
01 #include<iostream>
02 using namespace std;
03
04 void main()
05 {
06 int a;
07 unsigned int b;
08
09 a = -1;
10 b = -1;
11
12 cout << "a = " << a << '\n';
13 cout << "b = " << b << '\n';
14
15 b = 1111;
16
17 cout << "b = " << b << '\n';
18 }
```

signed 정수형 변수 a를 선언한다.

unsigned 정수형 변수 b를 선언한다.

unsigned 정수형 변수 b에 +1111을 입력한다.

01_변수의 기본

❖ 변수의 크기

- 변수의 크기는 변수 선언 시 정해주는 변수형에 따라 결정 된다.

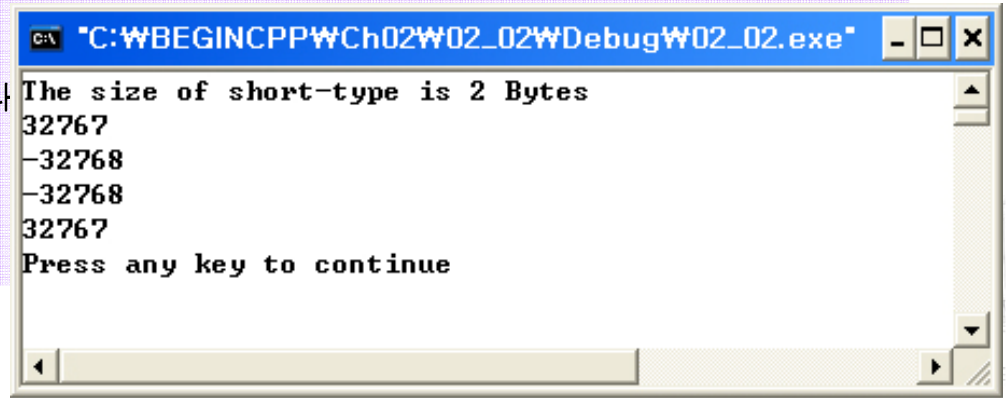
$$char < short < int \leq long \leq float < double < long\ double$$

❖ 오버플로우와 언더플로우

- 변수는 서로 자신만의 크기를 가지고 있다. 그런데 그 크기에 맞지 않게, 즉 변수에 담을 수 있는 한계가 넘는 수를 입력하면 변수는 제대로 동작하지 못한다.
- 오버플로우
  - 정해진 크기 중 가장 큰 값을 넘을 경우
- 언더플로우
  - 정해진 크기 중 가장 작은 값을 넘을 경우

[기본예제 2-2] 오버플로우

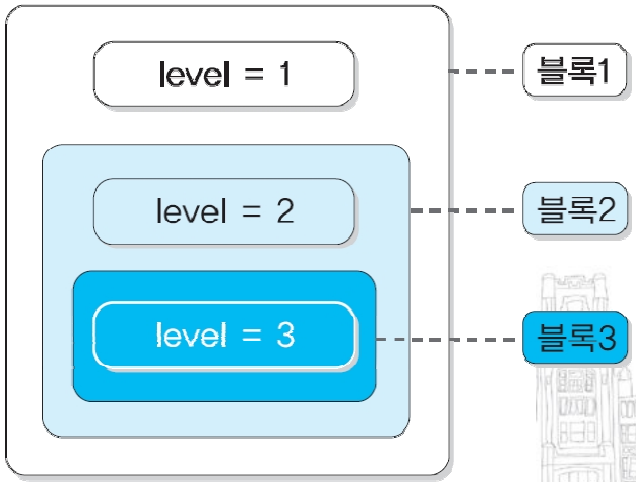
```
01 #include<iostream>
02 using namespace std;
03 void main()
04 {
05 short a = 0;                    signed short형 변수 a를 선언한다.
06
07 cout << "The size of short-type is " << sizeof(short) << " Bytes\n";
08
09 a = 32767;                      표시할 수 있는 숫자다.
10
11 cout << a << '\n';
12
13 a = 32768;                      오버플로우가 일어난다.
14
15 cout << a << '\n';
16
17 a = -32768;                     표시할 수 있는 숫자다.
18
19 cout << a << '\n';
20
21 a = -32769;                     언더플로우가 일어난다
22
23 cout << a << '\n';
24 }
```



01_변수의 기본

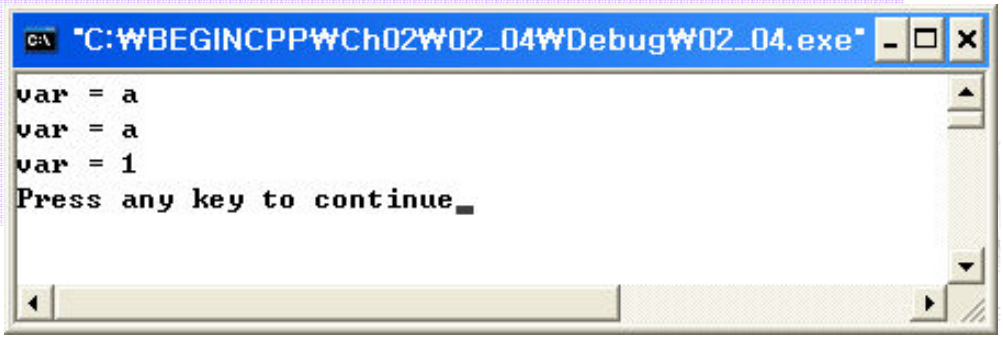
❖ 변수의 범위(scope)와 생명주기

- 변수가 선언되고 이용되고, 소멸되는 과정을 말한다.
- 선언된 변수가 유효한 범위를 말한다.
- 블록 안에서 선언된 변수는 블록 안에서만 유효하다.
  - 중첩된 블록 안에서 선언된 변수는 하위 블록(좀더 안쪽에 선언된 블록)들에서도 유효하다.
  - 소멸자를 호출하거나 인위적으로 변수를 해제하지 않는 이상, 먼저 생성된 변수가 나중에 소멸된다.
  - 기존에 존재하는 변수와 동일한 이름의 변수를 선언하면, 나중에 선언된 변수가 기존의 변수보다 우선한다.



[응용예제 2-4] 변수의 생명주기

```
01 #include<iostream.h>
02
03 void main()
04 {
05
06     int var = 1;
07     {
08
09         char var = 'a';
10         {
11
12             cout << "var = " << var << '\n';
13         }
14         cout << "var = " << var << '\n';
15     }
16     cout << "var = " << var << '\n';
17 }
```



01_변수의 기본

❖ 변수의 또 다른 분류

구분 방법	분류
선언 장소	지역변수, 전역변수
저장 장소	auto, static, register

- 지역 변수
  - **함수나 기타 블록 안에서 선언된 변수**
  - **그 블록 혹은 그 블록의 하위 블록에서만 유효한 변수**
- 전역 변수
  - **어디에도 속하지 않은 가장 바깥쪽 블록에서 선언된 변수**
  - **어디에서나 유효한 변수**
- 저장 장소
  - **auto – 자동 공간**
    - **register – 레지스터**
  - **static – 정적 공간**
  - **new / delete – 동적 공간**

02_구조체

❖ 원시 데이터 형과 복합 데이터 형

원시 데이터 형

Primitive data type 이라고도 하며, 컴파일러가 기본적으로 제공하는 데이터 형을 일컫는다.

복합 데이터 형

Compound data type이라고도 하며, 일반적으로 사용자가 직접 형을 정의하고, 그 객체를 값처럼 취급하거나 주소를 참조하여 사용되는 데이터 형을 말한다.

구조체나 클래스는 여기에 해당된다.

02_구조체

❖ 구조체의 기본 형태와 사용 예

```
struct 구조체명
{
변수_선언1;
변수_선언2;
....
(구조체_선언)
};
```

[기본 형태]

```
struct Student
{
int id;
float grade;
int gender;
};
```

[사용 예]

- 앞에 'struct' 라는 키워드를 사용하여, 구조체를 선언한다.
- 구조체가 가지는 변수들은 변수의 선언 방법과 같은 방법으로 선언된다.

[기본예제 2-5] Student 구조체의 사용 - 1

```
01 #include<iostream>
02 using namespace std;
03 struct Student
04 {
05         int id;
06         float grade;
07         int gender;
08 };
09
10 void main()
11 {
12         struct Student student;
13
14         cout << "학번을 입력하세요₩t";
15         cin >> student.id;
16
17         cout << "성적을 입력하세요₩t";
18         cin >> student.grade;
```

Student형 구조체 변수를 선언한다.

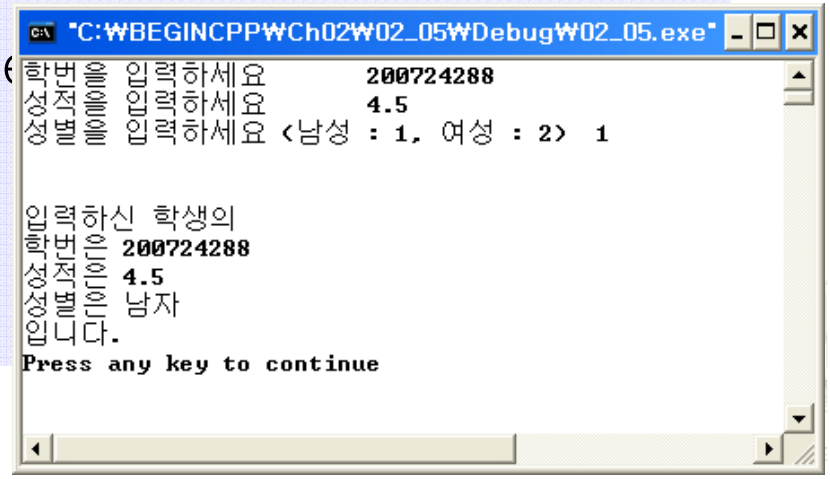
학번을 입력 받는다.

[기본예제 2-5] Student 구조체의 사용 - 2

```
19
20      cout << "성별을 입력하세요 (남성 : 1, 여성 : 2)₩t";
21      cin >> student.gender;
22
23      cout << "₩n₩n";
24      cout << "입력하신 학생의₩n";
25      cout << "학번은 " << student.id << '₩n';
26      cout.setf(ios::showpoint);
27      cout.precision(2);
28
29      cout << "성적은 " << (float)student.grade << '₩n';
30
31      cout << "성별은 " << (student.ge
    << '₩n';
32      cout << "입니다.₩n";
33 }
```

소수점을 표시하도록 설정한다.

소수점 1자리까지 표시하도록 설정한다.



03_네임스페이스(namespace)

❖ **namespace**

- 말 그대로 '이름이 유효한 공간' 이다.
- 특정 namespace안에서 선언된 이름 들은 그 namespace 블록 안에서만 유효하다.
- '::' 참조자를 이용하여 특정 namespace안의 변수에 접근이 가능하다.

❖ **예**

```
네임스페이스명::대상명
```

```
namespace 네임스페이스명
{
변수
클래스
함수
기타요소
.....
}
```

[부모 클래스]

```
namespace Elementary
{
        struct Student
        {
                int id;
                int age;
        };
}
```

[자식 클래스]

[기본예제 2-6] namespace의 사용 - 1

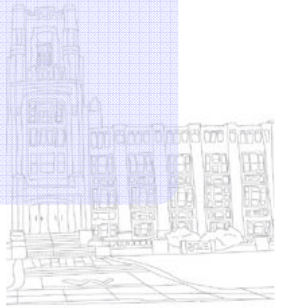
```
01 #include<iostream.h>
02
03 namespace Elementary
04 {
05         struct Student
06         {
07                 int id;
08                 int age;
09         };
10 }
11
12 namespace University
13 {
14         struct Student
15         {
16                 int id;
17                 float grade;
18         };
19 }
```

초등학생 정보 처리를 위한 네임스페이스를 정의한다.

초등학생 정보의 구조체를 정의한다.

대학생 정보 처리를 위한 네임스페이스를 정의한다.

대학생 정보의 구조체를 정의한다.



[기본예제 2-6] namespace의 사용 - 2

```
21 void main()
22 {
23
24 struct Elementary::Student eleStudent;
25
26 cout << "Enter Elementary student's ID₩n";
27 cin >> eleStudent.id;
28 cout << "Enter Elementary student's age₩n";
29 cin >> eleStudent.age;
30
31 struct University::Student uniStudent;
32
33 cout << "Enter University student's ID₩n";
34 cin >> uniStudent.id;
35 cout << "Enter University student's grade₩n";
36 cin >> uniStudent.grade;
37
38 cout << "Elementary student's" << '₩n';
39 cout << "₩t ID = " << eleStudent.id << '₩n';
40 cout << "₩t AGE = " << eleStudent.age << '₩n';
41 cout << "University student's" << '₩n';
42 cout << "₩t ID = " << uniStudent.id << "₩n";
43 cout << "₩t GRADE = " << uniStudent.grade << '₩n';
44 }
```

초등학생용 Student 구조체형 변수를 선언하고 값을 입력 받는다.

대학생용 Student 구조체형 변수를 선언하고 값을 입력 받는다.

서로 다른 Student에 값이 제대로 입력되었는지 출력해본다.

[기본예제 2-8] using의 사용 - 1

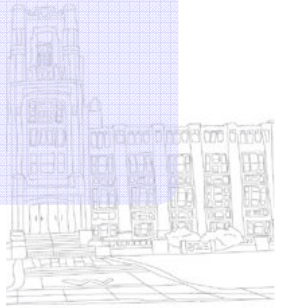
```
01 #include <iostream.h>
02
03 namespace Student
04 {
05 int id = 0;
06 int age = 8;
07 char grade = 'A';
08 }
09
10 void main()
11 {
12 cout << "ID = " << Student::id << '₩n';
13 cout << "AGE = " << Student::age << '₩n';
14 cout << "GRADE = " << Student::grade << '₩n';
15 }
```

대학생 Student namespace를 일일이 명시했다..

[응용예제 2-9] using의 사용 - 2

```
01 #include <iostream.h>
02
03 namespace Student
04 {
05 int id = 0;
06 int age = 8;
07 char grade = 'A';
08 }
09
10 using namespace Student;
11
12 void main()
13 {
14 cout << "ID = " << id << '₩n';
15 cout << "AGE = " << age << '₩n';
16 cout << "GRADE = " << grade << '₩n';
17 }
```

대학생 네임스페이스를 명시하지 않고 네임스페이스의 내용을 사용하겠다는 선언을 한다.



04_형 변환

❖ 암시적(Explicit) / 명시적(Implicit) 형 변환

암시적

특별히 명시해주지 않았지만, 서로 다른 형의 값 혹은 변수 간의 차이 때문에 자동적으로 일어나게 되는 형 변환을 말한다.

명시적

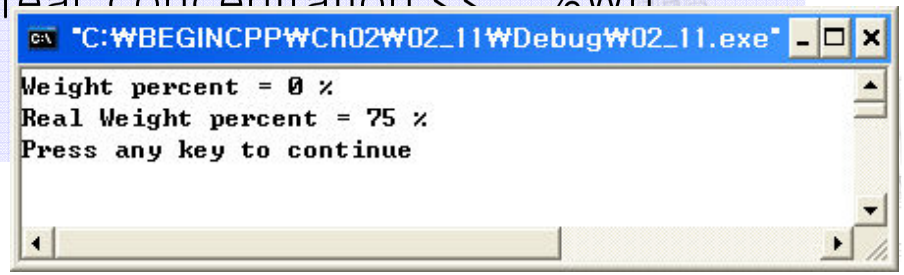
쉽게 암시적 형 변환의 반대라고 생각하면 되겠다. 즉, 명시적 형 변환은 우리가 그 형을 의도적으로 명시해 줌으로써, 형 변환을 수행하게 하는 것이다.

[기본예제 2-11] 명시적 형 변환의 사용

```
01 #include<iostream.h>
02
03 void main()
04 {
05 int salt = 30;
06 int water = 40;
07 float concentration = 0;
08 float real_concentration = 0;
09
10 concentration = (30 / 40) * 100;
11 cout << "Weight percent = " << concentration << " %₩n";
12
13 real_concentration = (float)30 / (float)40 * 100;
14 cout << "Real Weight percent = " << real_concentration << " %₩n";
15 }
```

대학생 형 변환 없이 정수 연산을 한다.

명시적 형 변환 후, 연산을 한다.



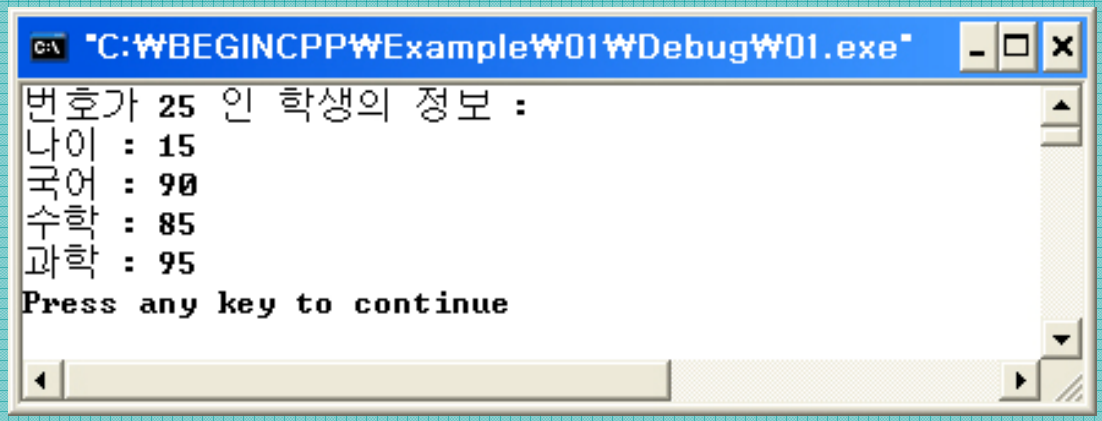
예제모음_1 학생 정보를 담는 구조체 설계

요구사항

① 학생의 정보를 담을 수 있는 구조체를 설계하라.
② 학생의 정보는 다음과 같이 구성된다.

ⓐ 번호: 25 ⓑ 나이: 15 ⓒ 국어성적: 90
ⓓ 수학성적: 85 ⓔ 과학성적: 95

실행결과



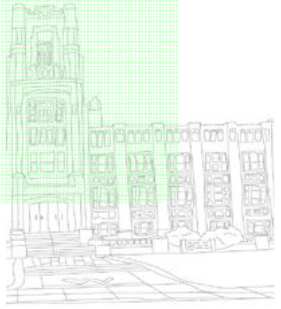
예제모음_1 소스

```
01 #include<iostream>
02 using namespace std;
03 struct Student
04 {
05 int number;
06 int age;
07 int kor;
08 int math;
09 int sci;
10 };
11
12 void main()
13 {
14 struct Student student;
15
```

학생 정보를 위한 구조체를 설계한다.

예제모음_1 소스

```
16 student.number = 25;
17 student.age = 15;
18 student.kor = 90;
19 student.math = 85;
20 student.sci = 95;
21
22 cout << "번호가 " << student.number << " 인 학생의 정보 : ₩n";
23 cout << "나이 : " << student.age << '₩n';
24 cout << "국어 : " << student.kor << '₩n';
25 cout << "수학 : " << student.math << '₩n';
26 cout << "과학 : " << student.sci << '₩n';
27 }
```



요약

변수

C++에서의 변수란 수학적 변수를 의미함과 동시에 그러한 변수를 담고 있는 '그릇' 까지 의미한다. 그리고 변수를 담고 있는 '그릇' 을 프로그래밍 환경과 연결시켜서 해석하면 그릇은 물리적인 메모리의 어딘가에 할당되어 있는 공간을 의미하게 된다.

구조체

구조체는 변수처럼 그 자체가 쓰이는 것이 아니라 변수형처럼 구조체형 변수 선언을 위한 형태를 정의해놓은 것이다. 아래는 구조체형 변수 선언의 일반적인 형태다.

namespace

namespace는 공동 작업 시 변수나 함수명 등 여러 모듈의 이름이 겹치는 것을 막기 위한, 말 그대로 '이름 공간' 이다. 따라서 서로 다른 이름 공간에, 동일한 이름의 변수가 있다고 할지라도 그 둘은 각각 다른 것이며, 각각의namespace 안에서만 그 이름이 유일한(unique) 것이 된다.

Thank You !

IT CookBook for Beginner, C++ 기초 2장 끝