조금 복잡한 프로그램

변영철 교수
(ycb@jejunu.ac.kr)

공부할 내용

- 지역변수, 전역변수
- 변수의 특징
- 함수를 이용한 프로그래밍

지역변수 정의

- 아래와 같이 2개의 **지역변수**(iX, iY)를 추가(정의)하자.

```
#include <stdio.h>

void main()
{
        int iX;
        int iY;

        iX = 2;
        iY = 3;

        int iResult = iX + iY;

        printf("두 개의 값을 더한 결과: %d₩n", iResult);
        getchar();
}
```

변수를 바깥으로 옮기기

- **지역변수**를 **전역변수**로 바꾸기

```
#include <stdio.h>

int iX;
int iY;

void main()
{
	iX = 2;
	iY = 3;

	int iResult = iX + iY;

	printf("두 개의 값을 더한 결과: %d₩n", iResult);
	getchar();
}
```

변수 정의? 선언?

- 변수 정의와 선언 (정만선알)
  - 정의(definition) : 변수를 만듬. 어디에?
  - 선언(declaration) : 변수가 있음을 알림. 누구에게?

(질문) 대한독립을 만방에 ?? 하노니...

지역변수와 전역변수 특징

- 얼마나 오랫동안 살까(life time)
  - (전역 변수) 프로그램이 실행될 때 만들어지고 프로그램이 종료될 때 같이 사라짐(오랫동안 존재).
  - (지역 변수) 함수가 실행될 때 만들어지고 함수가 끝나면 바로 사라짐(잠깐 있다가 사라짐)
- 어디에서 사용할 수 있을까(scope)
  - (전역 변수) 모든 곳 (모든 함수) 에서..
  - (지역 변수) 함수 안에서만 사용됨

지역변수와 전역변수 특징

구분	Life Time (수명)	Scope (사용범위)
전역변수	프로그램 실행시 프로그램 종료시 (오랫동안 살아있음)	프로그램 전체에서 (전체에서)
지역변수	함수 실행시 함수 종료시 (잠깐 살아있음)	함수 안에서만 (일부에서만)

지역변수를 전역변수로 바꾸는 이유

- 여기 저기에서 오랫동안 사용할 값을 저장하는 변수라면 전역변수로 정의
- 중요한 값을 저장하는 변수라면 전역변수로 정의

함수를 이용한 프로그래밍

- 수백 줄로 작성된 프로그램은?
  - 생각만 해도, 보기만 해도 머리가 아프다!
- 추상화(abstraction)
  - 복잡한 내용을 간단하게 줄여서 표현하는 것
  - 예) 어제 무엇을 했나요?
- 코드 추상화
  - 여러 줄의 코드를 간단히 표현하는 것

함수를 이용한 프로그래밍

```
#include <stdio.h>

int iX;
int iY;

void main()
{
    iX = 2;
    iY = 3

    int iResult = iX + iY;

    printf("두 값의 합: %d₩n", iResult);
    getchar();
}
```

```
#include <stdio.h>

int iX;
int iY;

void main()
{
    Assign(2, 3)

    int iResult = Add();

    printf("두 값의 합: %d₩n", iResult);
    getchar();
}
```

함수를 이용한 프로그래밍

```
#include <stdio.h>

int iX;
int iY;

void Assign (int x, int y)
{
    iX = x;
    iY = y;
}

void main()
{
    int iResult;

    Assign(2, 3);

    iResult = iX + iY;

    printf("두 개의 값을 더한 결과: %d₩n", iResult);
}
```

"코드 두 줄을 Assign 함수로 추상화한 것"

함수를 이용한 프로그래밍

```
#include <stdio.h>

int iX;
int iY;

void Assign(int x, int y)
{
    iX = x;
    iY = y;
}

int Add()
{
    return iX + iY;
}

void main()
{
    int iResult;

    Assign(2, 3);

    iResult = Add();

    printf("두 개의 값을 더한 결과: %d₩n", iResult);
}
```

왜 함수를 이용할까?

- 간단해 져서 보기에 편해졌다.
- 쉽게 읽을 수 있다.
- 프로그램 분석이 쉬다.
- 프로그램 고치기도 쉽다(유지 보수).
- 작성한 함수를 나중에 재사용 할 수도 있다.

프로그램 생산성이 좋아진다.

프로그램이 엄청 커지면?

- 짜다 보니 100개의 변수와 1000개의 함수로 이루어지면?
- 이것 저것 막 섞여 있다면?
- 프로그램 분석이 어려워진다.
- 프로그램 수정도 어려워진다.
- 따라서 담배만 는다?!

프로그램이 엄청 커지면?

- 수많은 변수와 함수 중 서로 관련된 것들을 모아서 다른 파일에 넣자.
- 서로 관련된 것끼리 다른 파일에 쪼개 넣자.

모듈별 분할 컴파일

구조화 프로그램과 모듈

- 구조화 프로그램
  - 실행 순서가 항상 위에서 아래로만 진행
  - 프로그램을 쉽게 읽을(이해할) 수 있음 : 가독성 (readability)이 좋아짐.
  - C 언어 -> 구조화 프로그래밍 언어라고 함.
- 모듈
  - 하나의 부품, 블록 -> 조립할 수 있는 단위
  - C언어에서는 하나의 파일을 모듈로 간주함. -> C언어를 모듈별 분할 컴파일이 가능한 언어라고 부름.
  - 함수도 하나의 모듈로 간주할 때도 있음.