Practice of Information Processing

（IMACU）

Fifth Lecture(Part 3)
Data type, structure, union

Makoto Hirota

Contents of this lecture

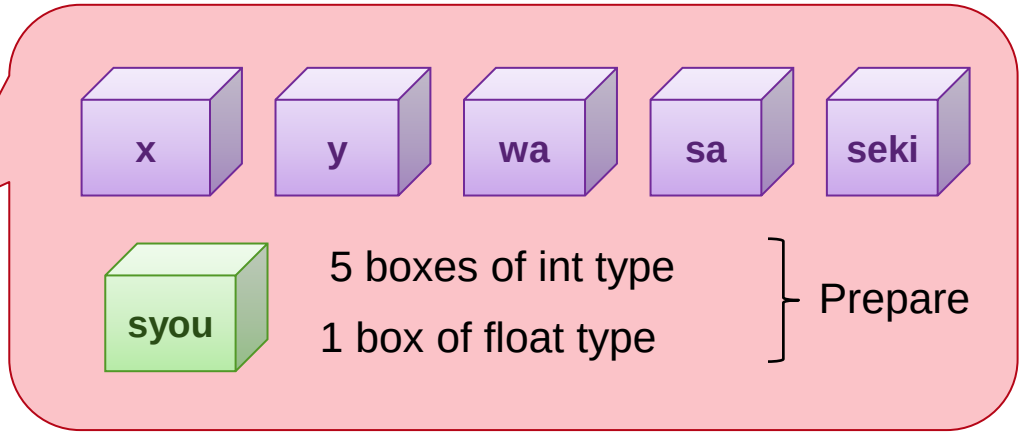
- Example answer of previous exercise
- Definition of function
  - What is function
  - Definition of self-made function
  - Prototype declaration
  - Role of function
- Function with no return value “procedure“
- Recursive call of function
- Data type
- What is structure?

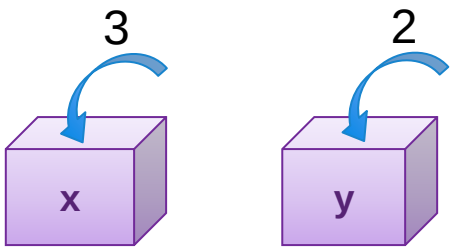
What is variables? (review)

- Variables are boxes for holding values in your program
  - The variable name is the name of the box
  - Declare the type of box (= variable type) to be used according to the range of numerical values to be handled
  - It is uncertain what is in the box after the declaration.
    (Be sure to explicitly substitute some value before using it = initialization)

❖ To be more precise, Box = memory area allocated for variable

```
#include <stdio.h>

int main(){

    /**** variable declaration****/
    int x, y;           /* int type */
    int wa, sa, seki;   /* int type */
    float syou;         /* real type */



    /**** processing contents****/
    /* assignment */
    x = 3;
    y = 2;

    return 0;
}
```





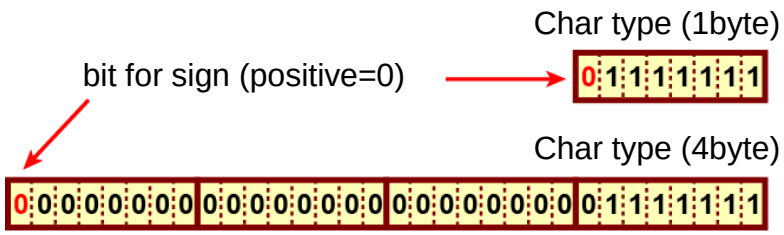
"=" means "assignment"

❖ Use "==" for the equality "=" in mathematics

Representative variable type（review）

Data type		byte	Range
char	Character type	1	−128～127 （If you use the ASCII code table, it will be treated as a character.）
short	Integer type	2	−32768 ～ 32767 （±300 thousand）
int	Integer type （32bit CPU）	4	−2147483648 ～ 2147483647 （±billion ） （16bit CPU: 2byte）
long	Long integer type （32bit CPU）	4	−2147483648 ～ 2147483647
long	Long integer type （64bit CPU）	8	−9223372036854775808～9223372036854775807 （"long long" type is 8digit integer, 32bit CPU environment）
float	Single precision real type	4	-3.40282×10^{38}～3.40282×10^{38} （Effective digit of approx. 7）
double	Double precision real type	8	-1.79769×10^{308}～1.79769×10^{308} (Effective digit of approx. 14)

$$127 = 2^6+2^5+2^4+2^3+2^2+2^1+2^0$$



※ The size of the data type depends on the execution environment (CPU). Normally, you don't need to be aware of it because the compiler has a definition that matches the CPU.

The significant digit also changes depending on whether the most significant bit is treated as a sign.

unsigned int type: 0 ～ 4294967295 (32bit)

Sign is not used

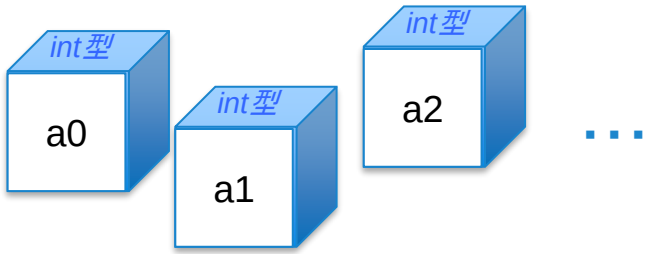
Array (review)

- The way managing many variables of the same type by assigning one name and consecutive numbers (index)

Ex) When you want to prepare 100 int type variables

<Previous variable declaration>

```
int a0, a1, a2, ···, a99;
a0=10;
a1=20;
```



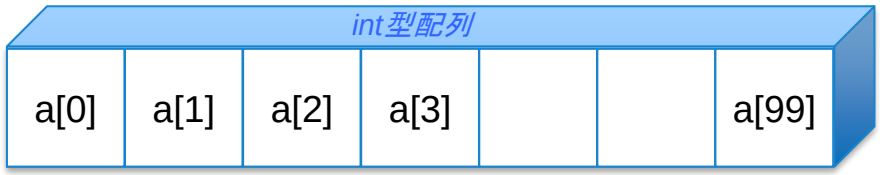
schematic of single variable

A separate data storage location (memory) is prepared for each variable

<Variable declaration using array>

```
int a[100];
a[0]=10;
a[1]=20;
```

Add [] (braces) after the variable name



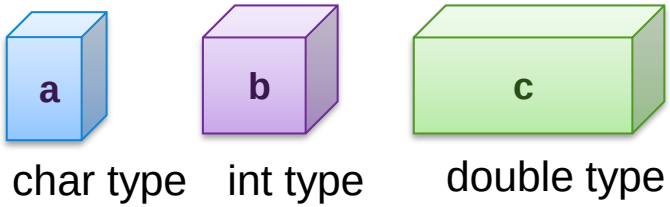
Schematic of array

Data in a continuous memory space
Storage space is secured

Data type summary

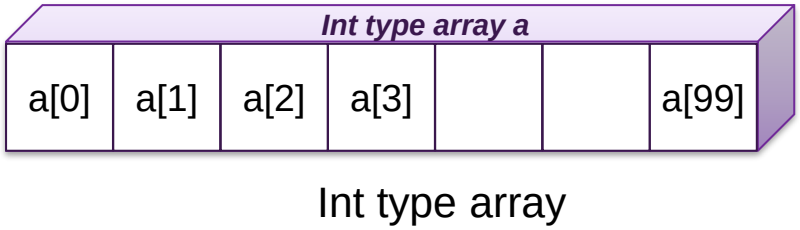
- Variable
  - Data type (box) whose size differs depending on the type of value to be entered
  - Type：character, integer, real

```
char a; int b;  double c;
```



- Array
  - Data type (box) in which variables of the same type are arranged

```
int a[100];
```

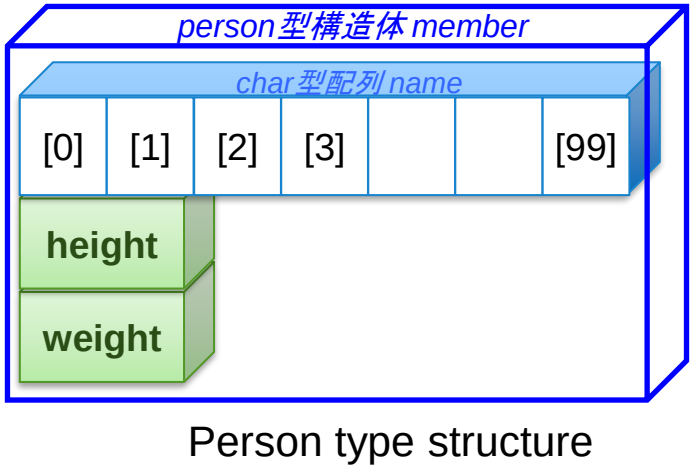


- Structure
  - Data type (box) that collects variables of different types
  - The name of the "type" that you can define

```
struct person member;
```

= Variable named "member" of "person" type structure

* Be careful not to confuse the structure name because it is a definition of "type" and not a variable name.



Structure

- Defined as a new type (= structure) to handle multiple different variables with one name

We can refer the member (component) by "Variable_name.(dot)member_name"

Defined outside the main function at the beginning of the program

```
struct person {
        char name[100];
        double height;
        double weight;
} ;   ←semicolon ";" is required
```

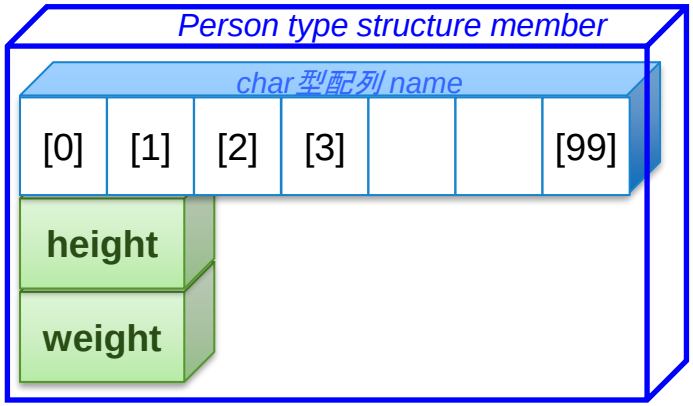
Can be declared as a variable type in a program

```
struct person member1, member2 ;   (Declare 2 variables)
```

struct person: Type name

member1, member2: variable variable

```
member1.height = 170.5;
member1.weight = 62.0;        (first person)

member2.height = 165.0;
member2.weight = 55.3;        (second person)
```



Person type structure

Ex: Program example using a structurestruct_ex.c


```c
#include <stdio.h>
#define NAME_LEN 100

/* definition of sturucture person*/
struct person{
    char name[NAME_LEN];
    double height;
    double weight;
};

int main(){

    /* variable declaration */
    struct person member;

    /* processing contents */
    /* data input */
    printf("Name? ");
    scanf ("%s",member.name);
    printf("Height? ");
    scanf ("%lf",member.height);
    printf("Weight? ");
    scanf ("%lf",member.weight);

    /* data display */
    printf("Name:   %s¥n",member.name);
    printf("Height: %.1f¥n",member.height);
    printf("Weight: %.1f¥n",member.weigth);

return 0;
}
```

- Structure definition
- Assignments and references to struct members

※ Since name is an array, & is not required for scanf

※ To correspond to the name with spaces, write the scan part as follows.

fgets(member.name, sizeof(member.name), stdin);

Supplement: Define an alias for the structure

Problem

If it is divided like struct person, it does not look like one variable name.

We just forget about the struct.

Sample program
- strcut_def1.c
- struct_def2.c
- struct_def3.c

- How to alias a type name: typedef

＜1＞ No alias

```
struct person{
    char name[NAME_LEN];
    double height;
    double weight;
};

int main(){

    //declaration
    struct person member;

    .
    .

}
```

Sample program struct_def1.c

＜2＞ Alias after declaration

```
struct person{
    char name[NAME_LEN];
    double height;
    double weight;
};
typedef struct person st_person;

int main(){

    //declaration
    st_person member;

    .
    .

}
```

struct_def2.c

＜3＞ Alias with declaration

```
typedef struct{
    char name[NAME_LEN];
    double height;
    double weight;
} st_person;

int main(){

    //declaration
    st_person member;

    .
    .

}
```

struct_def3.c

Supplement：Initialize when declaring a structure

- Initial values can be assigned with {,,} (bracket) at the time of declaration

Structure definition

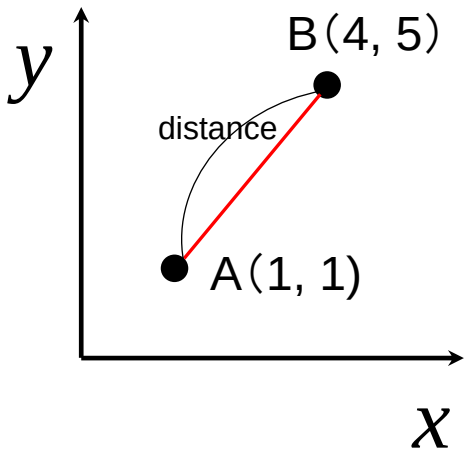
```
struct person{
    char name[NAME_LEN];
    double height;
    double weight;
};
typedef struct person st_person;
```

Declaration of structure

```
main(){

    st_person member = {“Ichiro”, 170, 50};

    printf("Name:   %s¥n",member.name);
    printf("Height: %.1f¥n",member.height);
    printf("Weight: %.1f¥n",member.weight);

}
```

Exercise 5-4 struct_point.c

- Create a program struct_point.c that finds the distance between two points in a two-dimensional plane.
  - For the coordinates of a certain point, define a structure Point having x and y coordinates as members, and use it for distance calculation.
  - Use double type coordinates
  - The square root uses the mathematical function sqrt ()
  - The square can be "x * x" or "pow (x, 2)" using the mathematical function pow ().
  - Coordinates are assigned directly in the main function. You may substitute (1.0, 1.0), (4.0, 5.0) at the time of declaration.



For mathematical functions
#include <math.h>
is necessary
(-lm for compilation)

Example of declaration:

```
struct Point a = {1.0, 1.0};
struct Point b = {5.0, 4.0};
```

  - Display the two coordinate values and the distance in the output.

Display example

```
a = (1.000000, 1.000000)
b = (5.000000, 4.000000)
distance = 5.000000
```

Array of structure

- Since a structure is a type of variable, it can also be an array.

```
main(){
     int i;
     struct person member[3];

     for (i=0; i<3; i++){
          scanf("%s", member[i].name);
          scanf("%lf", &member[i].height);
          scanf("%lf", &member[i].weight);
     }
}
```

```
main(){
     int i;
     st_person member[3];    (alias of structure name)

     for (i=0; i<3; i++){
          scanf("%s", member[i].name);
          scanf("%lf", &member[i].height);
          scanf("%lf", &member[i].weight);
     }
}
```

Exercise5-5: struct_array.c

- Rewrite the sample program struct_ex.c and create a program struct_array.c that inputs and displays the data of N people.
  - Define the number of people N as a macro (N = 3 is sufficient)
  - Also display the average height and weight of N people

Summary

- Example answer of previous exercise

- Definition of function
  - What is function
  - Definition of self-made function
  - Prototype declaration
  - Role of function
- Function with no return value “procedure“
- Recursive call of function

- Data type
- What is structure?

Next lecture

- Mechanism of computer
- Pointer
  - memory
    - Variable area in memory and address
  - Pointer to variable
  - Pointer to array
  - Pointer to structure