What is a variable?

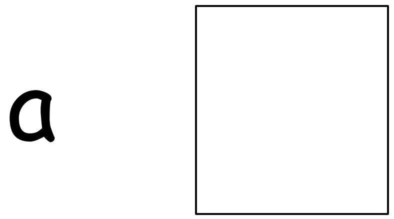
- Location in the memory where values can be stored
- It associates a name with value
- We can create a new variable by declaring it and then assigning value to it.

```
int a;
a = 2;
```

Variable declaration

```
#include <stdio.h>
int main()
{
    int a;
    a = 2;
   a = 100;
}
```

$$\begin{tikzcd}
 & \text{int a;} \arrow[dl] \arrow[dr] & \\
\text{Data-type of variable} & & \text{Name}
\end{tikzcd}$$



Variable assignment

```
#include <stdio.h>
int main()
{
    int a;
    a = 2;
   a = 100;
}
```

a = 2

$\downarrow$

assignment operator

a $\boxed{2}$

Variable update

```c
#include <stdio.h>
int main()
{
    int a;
    a = 2;
    a = 100;
}
```

a $\boxed{100}$

Data types in C

int: integer value (No decimal point)

```
int c;
c = 2;
```

float: real value (decimal points)

```
float c;
c = 4.3
```

char: characters

```
char c;
c = 'A'
```

Name of the variables

A variable name in C can be any valid **identifier.**

An identifier is a series of characters consisting of

letters (a-z, A- Z),

digits (0-9)

and underscores (_)

that does not begin with a digit.

Name of the variables

```
int a1;
int 1al
int A1;
int abcdef;
int _ab;
int A1;
```

Name of the variables

int a1;	valid
int 1al	invalid
int A1;	valid
int abcdef;	valid
int _ab;	valid : but avoid
int A1;	valid

Variable name: case sensitivity

C is **case sensitive**— uppercase and lowercase letters are different in C

int a1;

int A1;

a1, A1 two different variables

use small letters

Variable name: multi-word

int print_sum;

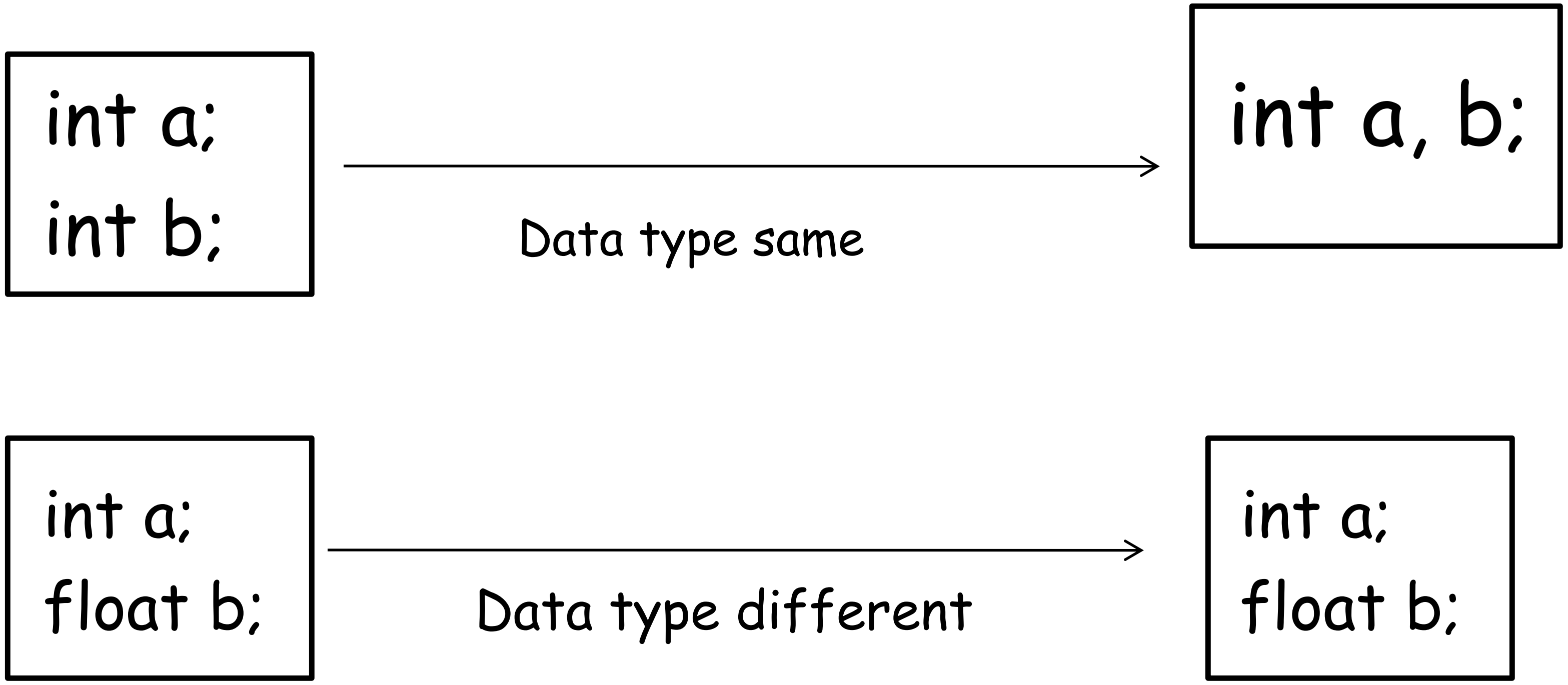
use '_' as the word separator

Variable initialization

```
int a;
a=2;
```

```
int a = 2;
```

Declaring multiple variables in single statement



```
int a;
int b;
```

```
int a, b;
```

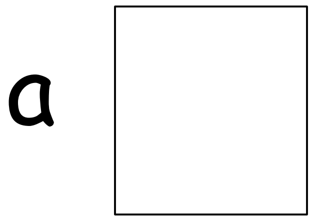
```
int a;
float b;
```

```
int a;
float b;
```

Declare variables before they are used

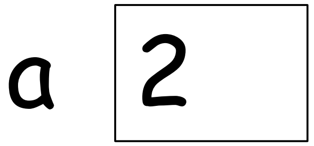
printing variables

```
#include <stdio.h>
int main()
{
    int a;
    a = 2;
    printf("%d", a);
}
```



scanning variables

```
#include <stdio.h>
int main()
{
    int a;
    a = 2;
    printf("%d", a);
}
```



Printing variables

```c
#include <stdio.h>
int main()
{
    int a;
    a = 2;
    printf("%d", a);
}
```

$a \ \boxed{2}$

```
2
```

Printing variables

```
printf("%d", a);
```

Name of the function for printing variables

Format specifier
- Depends on the data type of the variable you want to print
- Must be inside double quotes

See Datatype.pdf

Name of the variable you want to print

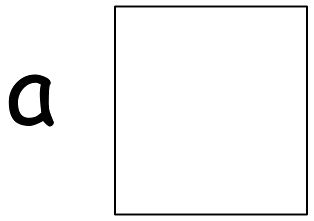
Printing multiple variables

```
#include <stdio.h>
int main()
{
    int a = 2;
    float b = 3.4;
    char c = 'A';
    printf("%d", a);
    printf("%f", b);
    printf("%c", c);
}
```

```
#include <stdio.h>
int main()
{
    int a = 2;
    float b = 3.4;
    char c = 'A';
    printf("%d, %f, %c", a, b, c);
}
```

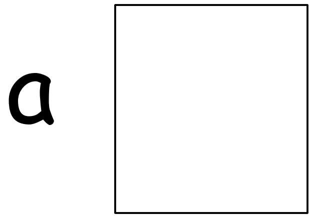
scanning variables

```
#include <stdio.h>
int main()
{
    int a;
    printf("Enter a:")
    scanf("%d", &a);
}
```



scanning variables

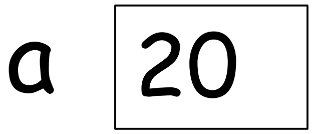
```
#include <stdio.h>
int main()
{
    int a;
    printf("Enter a:")
    scanf("%d", &a);
}
```



```
Enter a:
```

scanning variables

```
#include <stdio.h>
int main()
{
    int a;
    a = 2;
    printf("Enter a:")
    scanf("%d", &a);
}
```



```
Enter a:20
```

scanning variables

```
scanf("%d", &a);
```

Name of the function for scanning variables

Format specifier
- Depends on the data type of the variable you want to scan
- Must be inside double quotes

See Datatype.pdf

- Name of the variable you want to scan
- There must be & before it

Never forget the

&

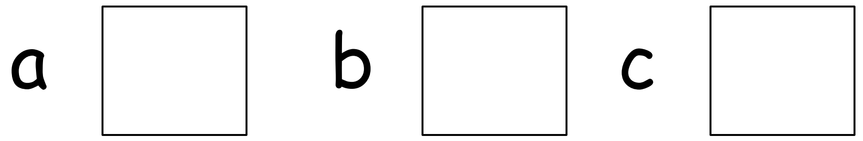
While scanning variables

Arithmetic Expression

```
#include<stdio.h>
int main() {
    int a, b, c;
    scanf("%d %d", &a, &b);
    c = a + b;
    printf("%d\n", c);
}
```

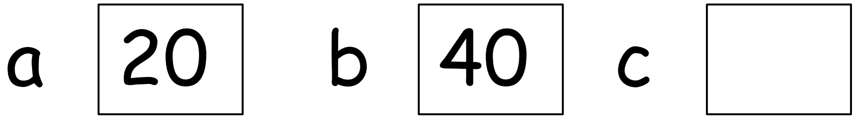
Arithmetic Expression

```
#include<stdio.h>
int main() {
    int a, b, c;
    scanf("%d %d", &a, &b);
    c = a + b;
    printf("%d\n", c);
}
```



Arithmetic Expression

```
#include<stdio.h>
int main() {
    int a, b, c;
    scanf("%d %d", &a, &b);
    c = a + b;
    printf("%d\n", c);
}
```



```
20 40
```

Arithmetic Expression

```
#include<stdio.h>
int main() {
    int a, b, c;
    scanf("%d %d", &a, &b);
    c = a + b;
    printf("%d\n", c);
}
```

a	b	c
20	40	60

```
20 40
```

Arithmetic Expression

```
#include<stdio.h>
int main() {
    int a, b, c;
    scanf("%d %d", &a, &b);
    c = a + b;
    printf("%d\n", c);
}
```

a $\boxed{20}$ b $\boxed{40}$ c $\boxed{60}$

```
20 40
60
```

Arithmetic Operators

Operation	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/
Remainder	%

- Addition and subtraction operates on all types (int, float, char)
- Multiplication and Division operate on int and float type
- Remainder only operates on int type

Algebraic and C Arithmetic Expressions

AE: $m = \dfrac{a+b+c+d+e}{5}$

C: $m = (a\ +\ b\ +\ c\ +\ d\ +\ e)/5$

AE: $y = mx + c$

C: $y\ =\ m * x\ +\ c$

Algebraic and C Arithmetic Expressions

AE: $a = pr \bmod q + \frac{w}{x} - y$

C: $a = (p * r) \% q + (w/x) - y$

AE: $y = ax^3 + bx^2 + cx + d$

C: $y = a * x * x * x + b * x * x + c * x + d$

Printing floats with specific precision

```
float a = 3.1415987666;
printf("%f", a);
```

```
3.141599
```

Default precision = 6

Printing floats with specific precision

```
float a = 3.1415987666;
printf("%.1f", a);
```

```
3.1
```

```
float a = 3.1415987666;
printf("%.4f", a);
```

```
3.1415
```

Compound Assignment Operator

```
a = a + b;
```

```
a += b;
```

```
a = b + c;
```

```
a = b + c;
```

Compound Assignment Operator

```
a = a - b;
a = a * b;
a = a / b;
a = a % b;
```

```
a -= b;
a *= b;
a /= b;
a %= b;
```

Increment Operator

$$\begin{tikzcd}
\text{a++;} \arrow[d] & & \text{++a;} \arrow[d] \\
\text{post-increment (post-fix)} & & \text{pre-increment (pre-fix)}
\end{tikzcd}$$

Decrement Operator

```
a--;
```

post-decrement
(post-fix)

```
--a;
```

pre-decrement
(pre-fix)