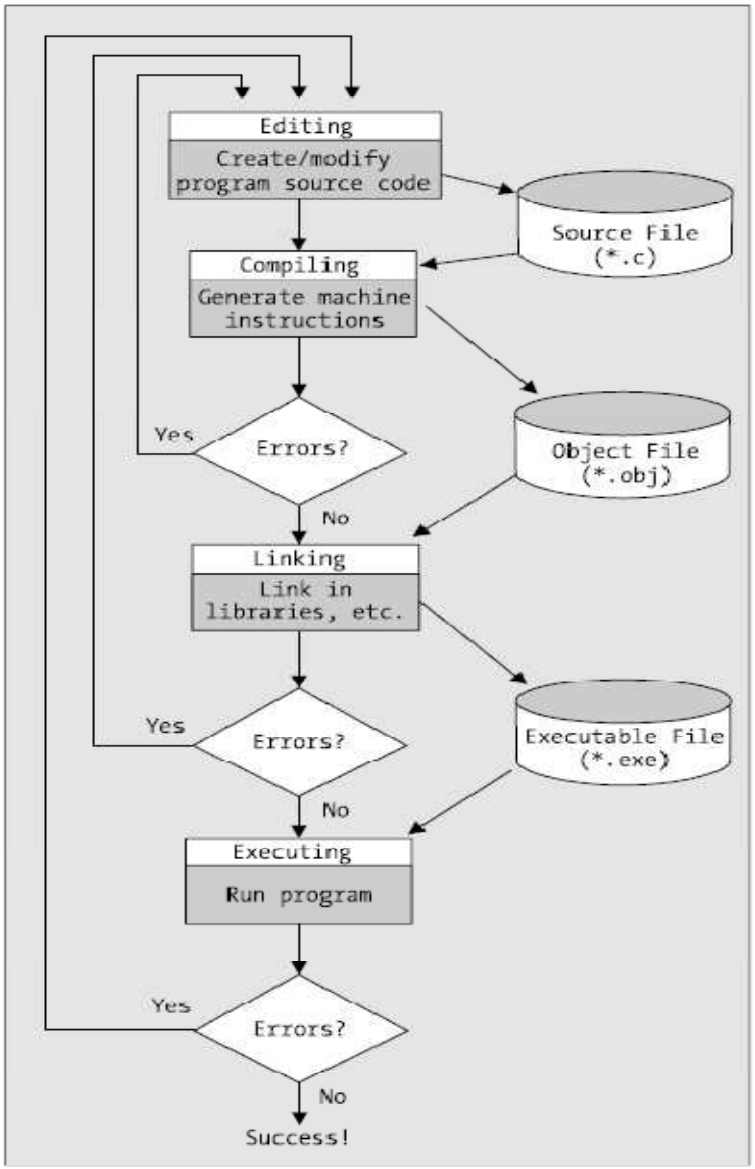
Variables, Data Type and Operators

Review

How To
Program
In
C



C Structure

```
#include <stdio.h>
#include <conio.h>
main()
{
  printf(“Programming is easy!!”);
}
other_function(){
 statement ;
}
```

Escape Character

Escape Sequence	Description
\n	Represents a newline character
\r	Represents a carriage return
\b	Represents a backspace
\f	Represents a form-feed character
\t	Represents a horizontal tab

Escape Sequence	Description
\v	Represents a vertical tab
\a	Inserts a bell (alert) character
\?	Inserts a question mark (?)
\"	Inserts a double quote (")
\'	Inserts a single quote (')
\\	Inserts a backslash (\)

Sample

```
#include <stdio.h>
#include <conio.h>
main()
{
  printf(“Programming is easy!!”);
  printf (“\n”);
  printf(“and fun”);
}
```

Preprocessor Directive

- stdio.h
  - printf()
  - scanf()
- conio.h
  - getch()
  - getche();
- stdlib.h
- math.h
- etc

Question ?

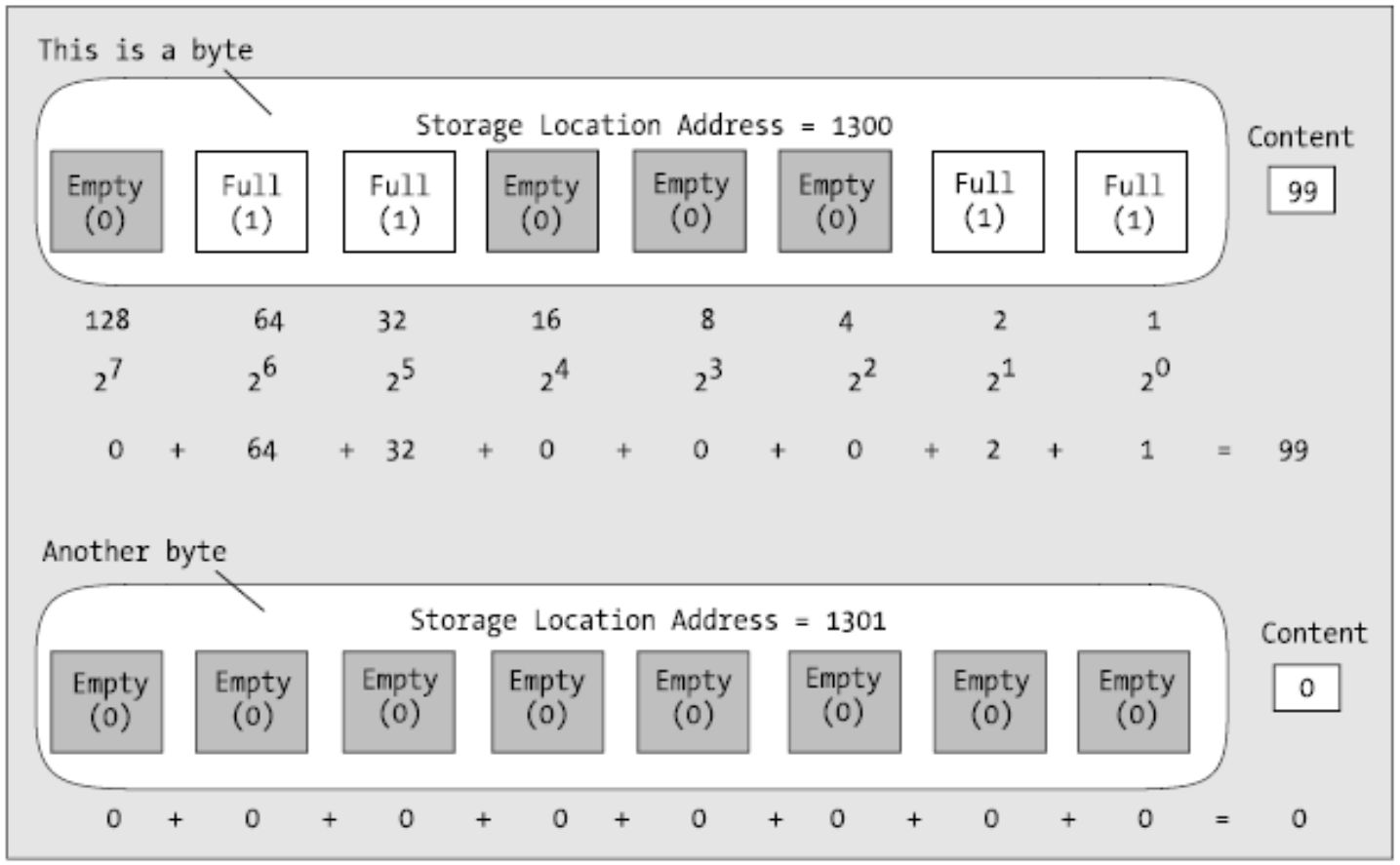
Today…

- Memory in computer
- Variable
- Data Type

Memory

- When your program is running, the storage place is main memory, or the random access memory (**RAM**),
- Another memory **ROM** : Read Only Memory (**ROM**) cannot be changed. The information contained in ROM was put there when the machine was manufactured, eq. basic input/output system (**BIOS**) in a PC
- You can think of your computer's RAM as an ordered sequence of boxes, it contains 1 or 0 (**bit**),

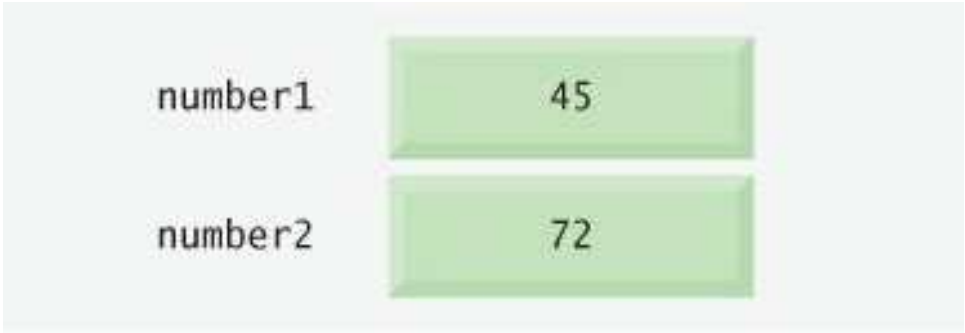
Memory (cont.)


This is a byte
Storage Location Address = 1300
Content
99
Empty (0)
Full (1)
Full (1)
Empty (0)
Empty (0)
Empty (0)
Full (1)
Full (1)
128 64 32 16 8 4 2 1
2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
0 + 64 + 32 + 0 + 0 + 0 + 2 + 1 = 99
Another byte
Storage Location Address = 1301
Content
0
Empty (0)
Empty (0)
Empty (0)
Empty (0)
Empty (0)
Empty (0)
Empty (0)
Empty (0)
0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0

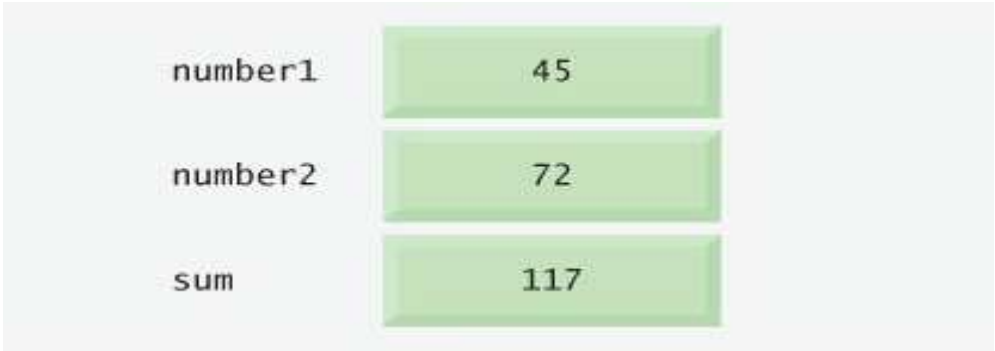
Variable

- A **place to store** an item of data that can vary in a program,
- Every variable has a name, it's called **Variable Name,**
- Use that **name** to refer to that place in memory to retrieve what it contains or store a new data value there,

Sample



After Calculating



Variables Declaration

- DATA_TYPE variable_name;
- Eq.

```
int a;
float f;
double d;
```

Assigning a Value To Variables

- Use = (assignment)
- Eq.

```
int a = 1;
float f = 2.0;
double d;
d = 3.0;
```

Role (Variable Name)

- Can't begin with a digit,

```
int 8a; (invalid)
int a8; (valid)
```

- Can't include any other characters besides letters, underscores, and digits,

```
int a-8; (invalid)
double d!; (invalid)
char segi tiga; (invalid)
```

- Variable names are case sensitive

```
int a; Not Equals with int A;
```

Role (cont.)

- Cannot use **keywords**

auto	for	struct
break	goto	switch
case	if	typedef
char	inline	union
const	int	unsigned
continue	long	void
default	register	volatile
do	restrict	while
double	return	_Bool
else	short	_Complex
enum	signed	_Imaginary
extern	sizeof	
float	static	

Excercise

- int x; (valid/invalid)
- int double; (valid/invalid)
- float jumlah pajak; (valid/invalid)
- float jumlah_pajak; (valid/invalid)
- Float jumlah_pajak; (valid/invalid)
- float jumlah^pajak; (valid/invalid)
- char 90a; (valid/invalid)
- char c_9; (valid/invalid)
- bool tanda; (valid/invalid)

Data Type

Type	Typical Size in Bits	Minimal Range
char	8	−127 to 127
unsigned char	8	0 to 255
signed char	8	−127 to 127
int	16 or 32	−32,767 to 32,767
unsigned int	16 or 32	0 to 65,535
signed int	16 or 32	Same as **int**
short int	16	−32,767 to 32,767
unsigned short int	16	0 to 65,535
signed short int	16	Same as **short int**
long int	32	−2,147,483,647 to 2,147,483,647
long long int	64	$-(2^{63}-1)$ to $2^{63}-1$ (Added by C99)
signed long int	32	Same as **long int**
unsigned long int	32	0 to 4,294,967,295
unsigned long long int	64	$2^{64}-1$ (Added by C99)
float	32	1E−37 to 1E+37 with six digits of precision
double	64	1E−37 to 1E+37 with ten digits of precision
long double	80	1E−37 to 1E+37 with ten digits of precision

Table 2 -1. *All Data Types Defined by the C Standard*

Printing data

```
#include <stdio.h>
#include <conio.h>
main()
{
  int a = 1;
  printf("Nilai a adalah %d ",a);
}
```

Printing Data

```
#include <stdio.h>
#include <conio.h>
main()
{
  int a,b;
  a = 1;
  b = 2;
  printf("Nilai a adalah %d dan nilai b adalah %d
    ",a,b);
```

Conversion Format

- %d : int,long,short,etc.
- %f : float,double,long double.
- %c : char.

Question ?

Exercise

- How to print this value

```
#include <stdio.h>
#include <conio.h>
main()
{
  float a = 10.0;
  printf(?);
}
```

Exercise

- How to print this value

```
#include <stdio.h>
#include <conio.h>
main()
{
  float a,b;
  a = 10.00;
  b = 5.00;
  Nilai a adalah 10.00 dan b adalah 5.00
}
```

Operator

- Arithmetic
- Logical
  - Relational
  - Equality

Arithmetic Operator

Table 2-1. *Basic Arithmetic Operators*

Operator	Action
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Arithmetic Operator (cont)

- Post increment

  a++ ~ a = a + 1

- Pre increment

  ++a ~ a += 1

- Post decrement

  a-- ~ a = a - 1

- Pre decrement

  --a ~ a -= 1

Arithmetic Operator (cont)

- Operator precedence

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
* / %	Multiplication Division Modulus	Evaluated second. If there are several, they are evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

Logical Operator

Standard algebraic equality or relational operator	C++ equality or relational operator	Sample C++ condition	Meaning of C++ condition
Relational operators			
>	>	`x > y`	`x` is greater than `y`
<	<	`x < y`	`x` is less than `y`
≥	>=	`x >= y`	`x` is greater than or equal to `y`
≤	<=	`x <= y`	`x` is less than or equal to `y`
Equality operators			
=	==	`x == y`	`x` is equal to `y`
≠	!=	`x != y`	`x` is not equal to `y`

Question ?

Exercise : How to write in C?

- Algebra : $m = \frac{a + b + c + d}{5}$
- Algebra : $y = mx + c$
- $F = ma$
- $y = ax^2 + bx + c$
- Other…

Answer

$$ax^2 + bx + c$$

```
Step 1.   y = 2 * 5 * 5 + 3 * 5 + 7;     (Leftmost multiplication)
              2 * 5 is 10
Step 2.   y = 10 * 5 + 3 * 5 + 7;        (Leftmost multiplication)
              10 * 5 is 50
Step 3.   y = 50 + 3 * 5 + 7;            (Multiplication before addition)
                  3 * 5 is 15
Step 4.   y = 50 + 15 + 7;               (Leftmost addition)
              50 + 15 is 65
Step 5.   y = 65 + 7;                    (Last addition)
              65 + 7 is 72
Step 6.   y = 72                         (Last operation—place 72 in y)
```

Good Programming Practice

- Using **redundant parentheses** in complex arithmetic expressions can make the expressions clearer

$$y = ax^2 + bx + c$$

y = (a*x*x) + (b * x) + c

Evaluation

1. Sebuah program perhitungan **luas segi tiga** memiliki variabel **alas** dan **tinggi**. Bilangan yang digunakan adalah bilangan pecahan. Tuliskan programnya hingga output luas lingkaran tercetak di layar monitor.Ctt.luas lingkaran berupa bilangan pecahan.
2. Tulis program untuk menghitung **luas lingkaran.**