OBJECT ORIENTED PROGRAMMING IN C++

Data Type

Professor 최학남

xncui@inha.ac.kr

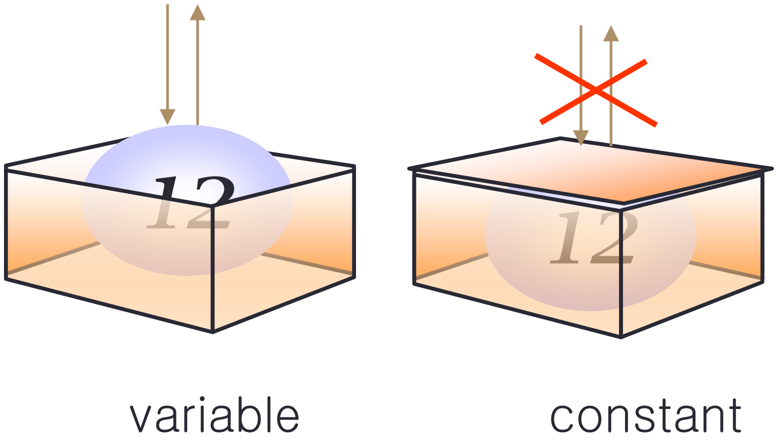
Office: high-tech 401

Contents

- Variable and Constant
- Data types
  - ✓ Integer type
  - ✓ Floating-point type
  - ✓ Character type
- Symbolic constant
- Underflow and overflow

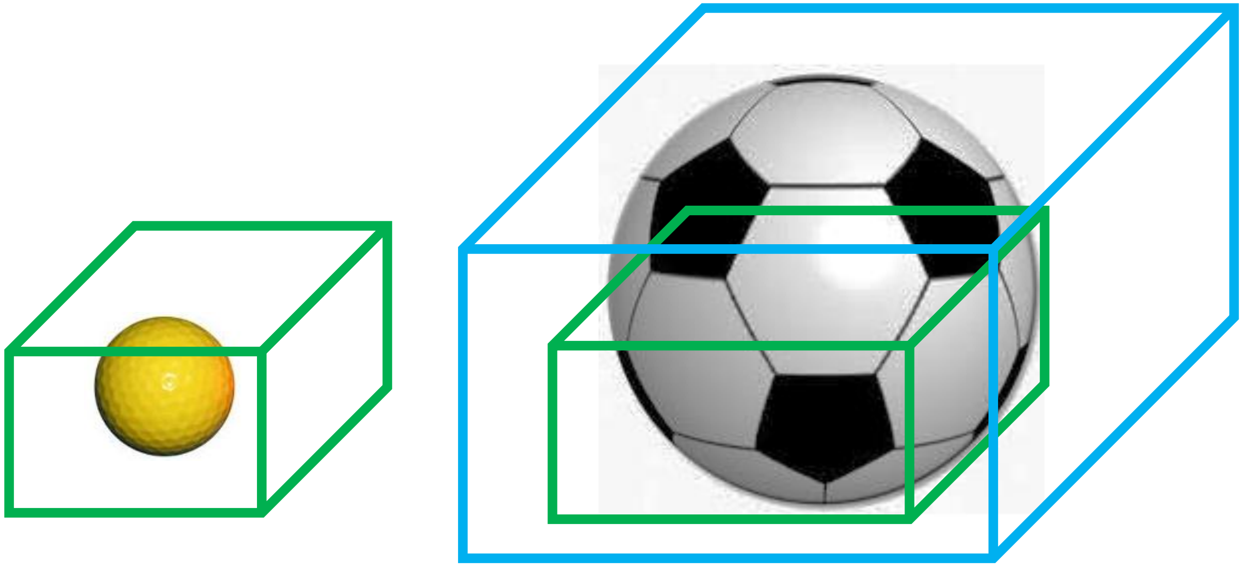
Variable and constant

- Variable
  - ✓Can change the allocated value
- Constant
  - ✓Can not change the allocated value



Data type

- Why need data type?
  - ✓Integer data : 100
  - ✓Real data : 3.14

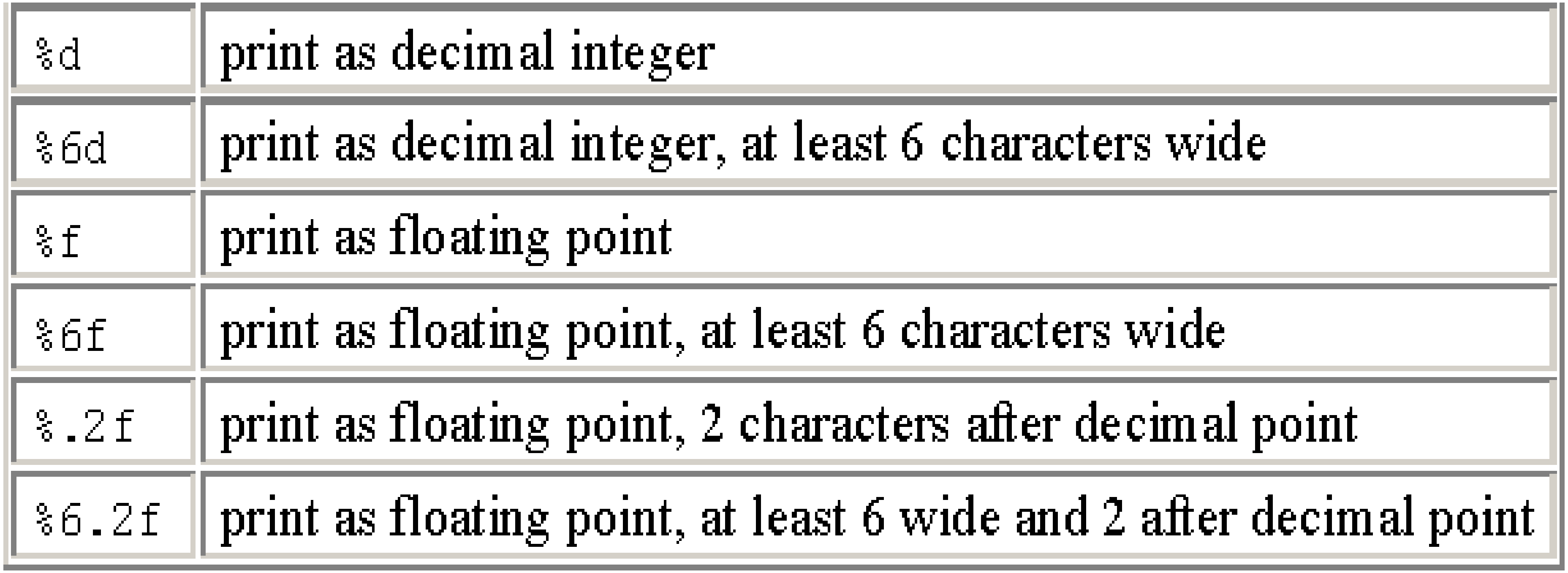


Data type table

Data type			Byte	Scope
Integer	signed	short	2	-32768~32767
		int	4	-2147483648~2147483647
		long	4	-2147483648~2147483647
		long long	8	-9,223,372,036,854,775,808~9,223,372,036,854,775,807
	unsigned	unsigned short	2	0~65535
		unsigned int	4	0~4294967295
		unsigned long	4	0~4294967295
Character	signed	char	1	-128~127
	unsigned	unsigned char	1	0~255
Floating-point		float	4	-3.4e-38 ~3.4e+38
		double	8	-1.7e-308 ~1.7e+308

32 bit compiler

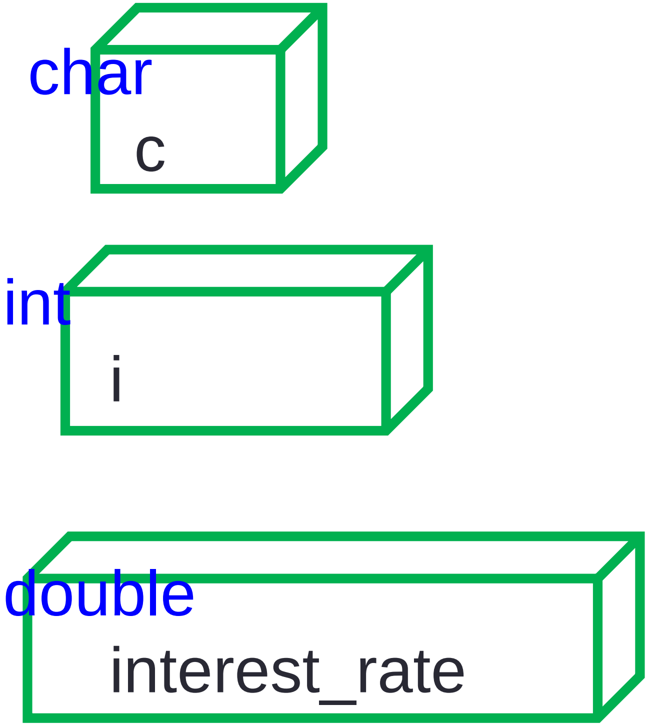
Format for **printf()** function



%d	print as decimal integer
%6d	print as decimal integer, at least 6 characters wide
%f	print as floating point
%6f	print as floating point, at least 6 characters wide
%.2f	print as floating point, 2 characters after decimal point
%6.2f	print as floating point, at least 6 wide and 2 after decimal point

Variable declaration

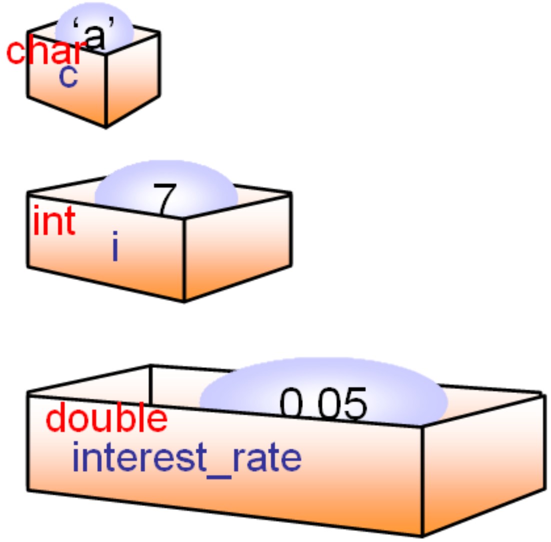
- data type variable name;
- Example
  - ✓char c;
  - ✓int i;
  - ✓double interest_rate;
  - ✓int height, width;



Variable declaration

- Initialize the variable

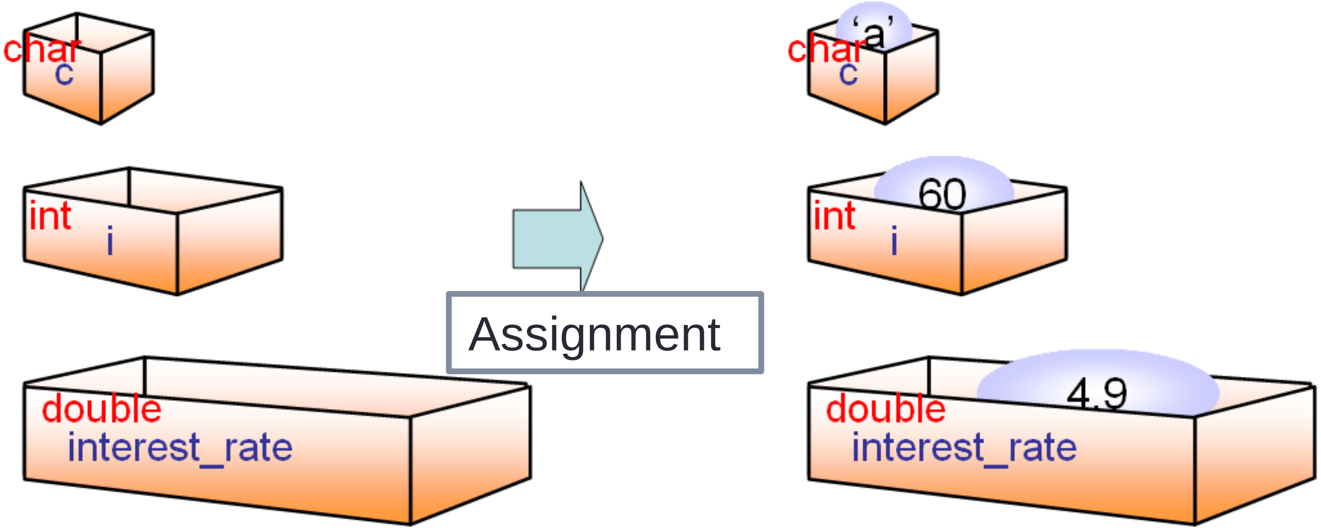
```
char c=‘a’;
int i=7;
double interest_rate=0.05;
```



Variable declaration

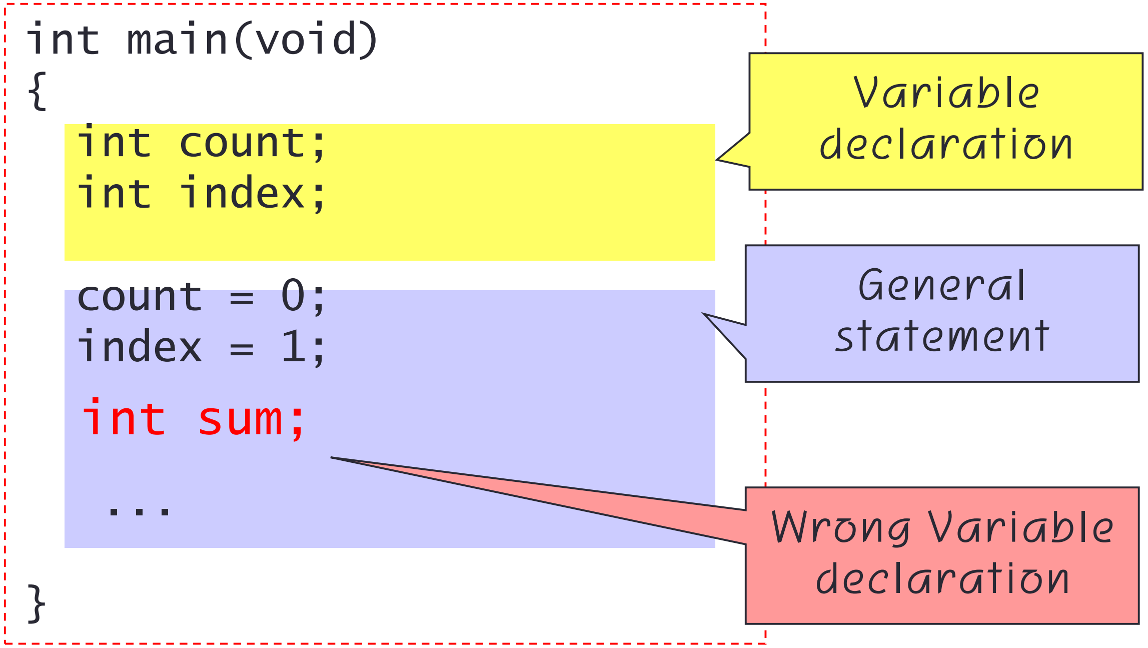
```
char c;                    // declared the character type variable c
int i;                     // declared the integer type variable i
double interest_rate;//declared the double type variable interest_rate


c = 'a';                   // assign the ‘a’ to the variable c
i = 60;                    // assign the 60 to the variable I
interest_rate = 4.9;     // assign the 4.9 to the variable interest_rate
```



Variable declaration

- ➢ Position of the variable declaration
  - ✓Located first part of the function



```
int main(void)
{
   int count;
   int index;

   count = 0;
   index = 1;
   int sum;

    ...

}
```

Variable declaration

General statement

Wrong Variable declaration

Integer type

- signed
  - ✓The first bit is for sign

$$-2^{31},\ldots,-2,\ -1,\ 0,\ 1,2,\ldots,\ 2^{31}-1$$

$$-2147483648 \leqq n \leqq +2147483647$$

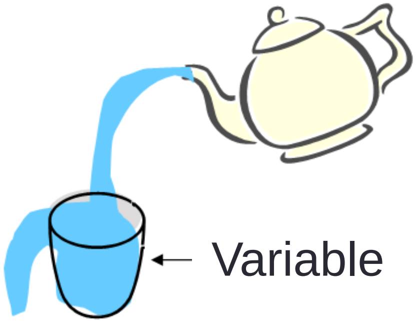
- unsigned
  - ✓using full bit

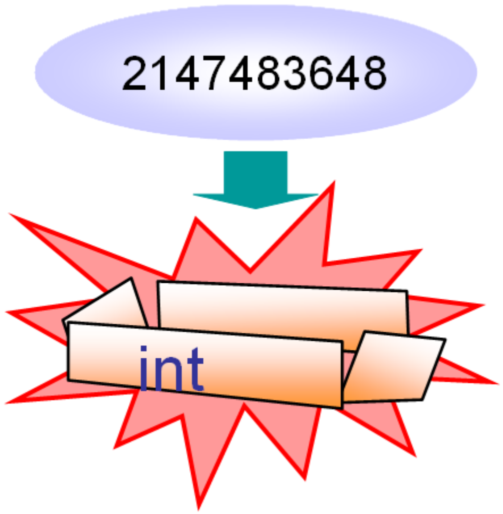
$$0,1,2,3,\ldots,2^{32}$$

$$0\leq n \leq 4294967294$$

Integer type

- Overflow

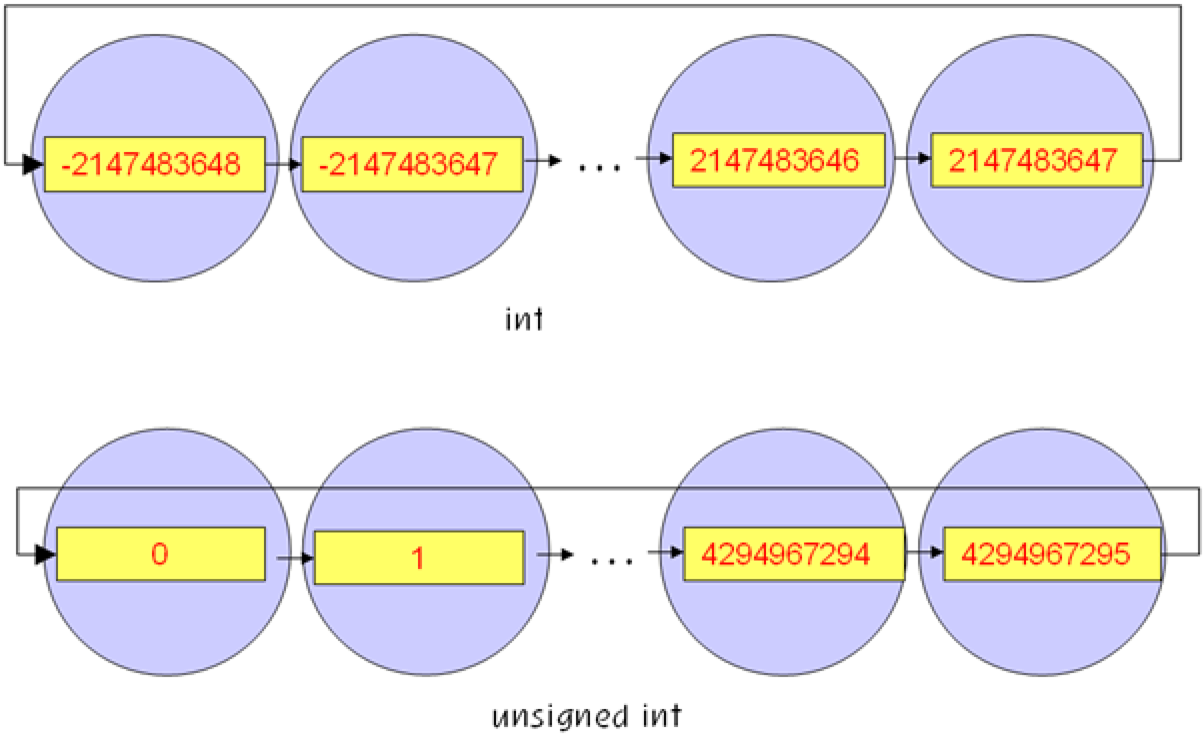




overflow

Integer type

- Overflow



Ex1-1: size of data type

```
/* compute the size of the data type*/
#include <stdio.h>

int main(void)
{
        short year = 0;             // initialization
        int sale = 0;               // initialization
        long total_sale = 0;        // initialization .

        year = 10;                  //  assignment the value careful the
        sale = 200000000;           //
        total_sale = year * sale;  //

        printf("total_sale  = %d \n", total_sale);

        printf(“size of short type %d byte \n", sizeof(short));
        printf(" size of int type %d byte \n", sizeof(int));
        printf(" size of long type %d byte \n", sizeof(long));

        return 0;
}
```

Ex 1-2: overflow

```
#include <stdio.h>
int main(void)
{
        int x;
        unsigned int y;

        x = 2147483647;
        printf("x = %d\n",x);
        printf("x+1 = %d\n",x+1);
        printf("x+2 = %d\n",x+2);
        printf("x+3 = %d\n",x+3);


        y = 4294967295;
        printf("y = %u\n",y);      // unsigned data type using %u
        printf("y+1 = %u\n",y+1);
        printf("y+2 = %u\n",y+2);
        printf("y+3 = %u\n",y+3);
       return 0;
}
```

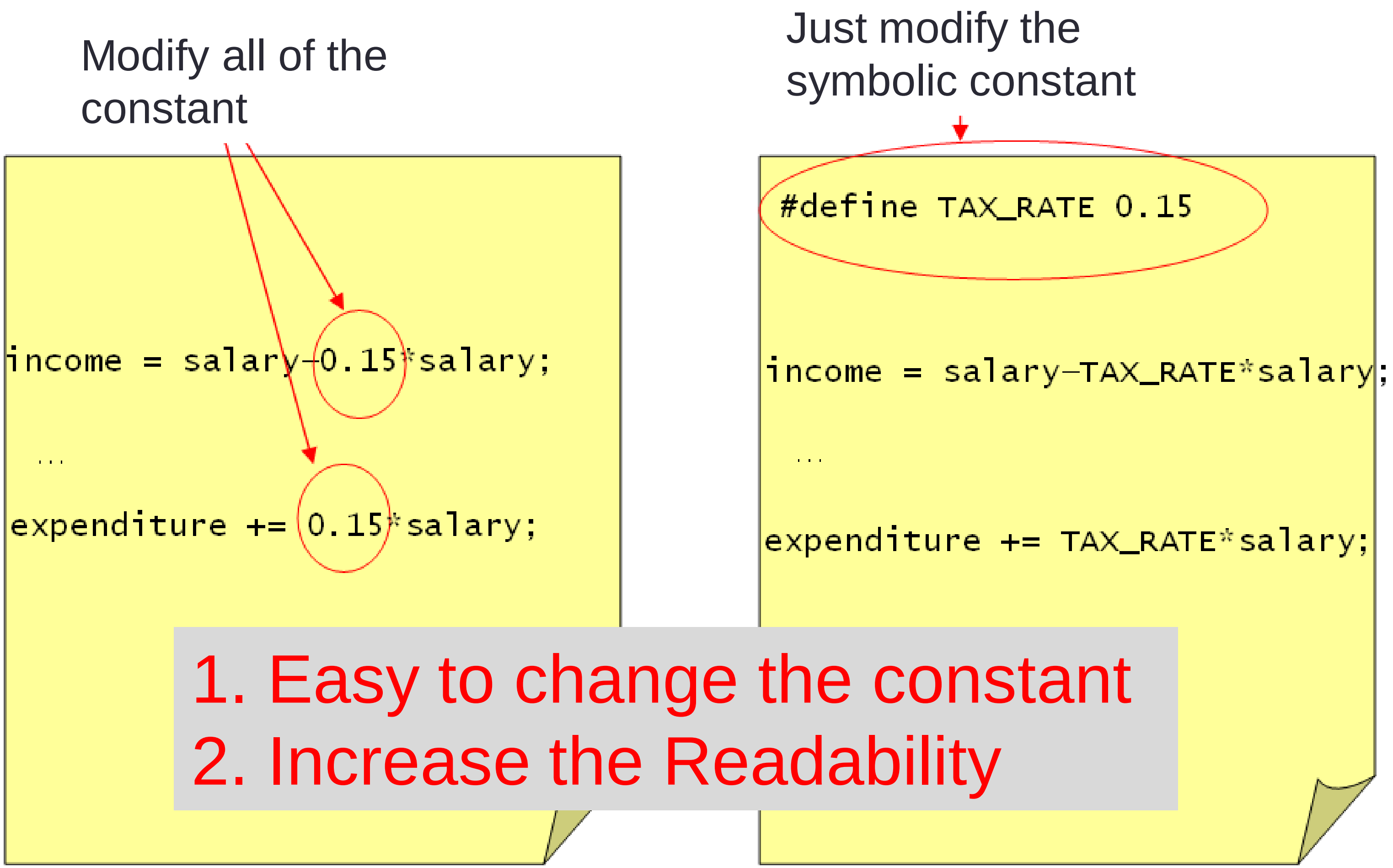
```
x = 2147483647
x+1 = -2147483648
x+2 = -2147483647
x+3 = -2147483646
y = 4294967295
y+1 = 0
y+2 = 1
y+3 = 2
```

Constant

- Symbolic constant
  - ✓#define N 1000

- Constant keyword : const
  - ✓const int N = 1000;

Advantage of Symbolic constant



Modify all of the constant

```
income = salary-0.15*salary;

 ...

expenditure += 0.15*salary;
```

Just modify the symbolic constant

```
#define TAX_RATE 0.15


income = salary-TAX_RATE*salary;

 ...

expenditure += TAX_RATE*salary;
```

1. Easy to change the constant
2. Increase the Readability

Example 2: symbolic constant

```
/* example for symbolic constant*/
#include <stdio.h>
#define PI 3.141592 // symbolic constant

int main(void)
{
        float radius, area, circumference;         //

        printf(“insert the radius:");              //
        scanf("%f", &radius);                      // get the value from keyboard

        area = PI * radius * radius;               // calculate the area
        circumference = 2.0 * PI * radius;         //calculate the circumference

        printf(“radius = %f.\n", radius);
        printf(“circle area = %f, circumference = %f \n", area, circumference);

        return 0;
}
```

Example 3: const keyword

```
/* symbolic constant  using const keyword*/
#include <stdio.h>


int main(void)
{
        const double TAX_RATE = 0.15;      // symbolic constant for tax rate
        double income, salary;             //


        printf(“insert your salary:");          //
        scanf("%lf", &salary);              // double data type using %lf

        income = salary - TAX_RATE * salary;     // calculate the net income
        printf(“net income : %lf\n", income);    //  print the net income


        return 0;
}
```

Example 4: const keyword

```
/* symbolic constant  using const keyword*/
#include <stdio.h>


int main(void)
{
        const double TAX_RATE = 0.15;      // symbolic constant for tax rate
        double income, salary;             //


        printf(“insert your salary:");          //
        scanf("%lf", &salary);             // double data type using %lf

        income = salary - TAX_RATE * salary;      // calculate the net income
        printf(“net income : %lf.\n", income);    //  print the net income

         TAX_RATE = 0.20;
        income = salary - TAX_RATE * salary;      // calculate the net income
        printf(“net income : %lf\n", income);    //  print the net income


        return 0;
}
```

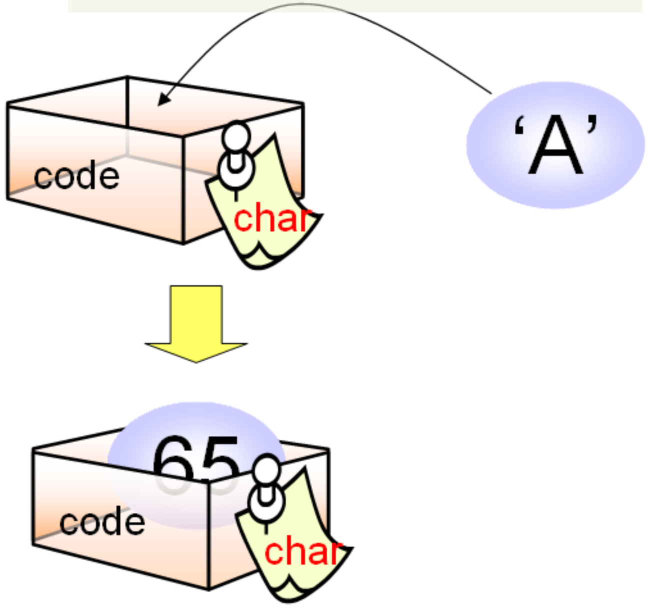
Character data type

- char data type store the character

```
char c;
char answer;
char code;
```

- American Standard Code for Information Interchange(ASCII)
  - ✓Explain the English alphabet using 8 bit
  - ✓!=33, ‘A’ = 65, ‘B’ = 66, ‘a’=97

```
code = 65;          // ‘A’
code = ‘A’;
```



Example 5: ASCII

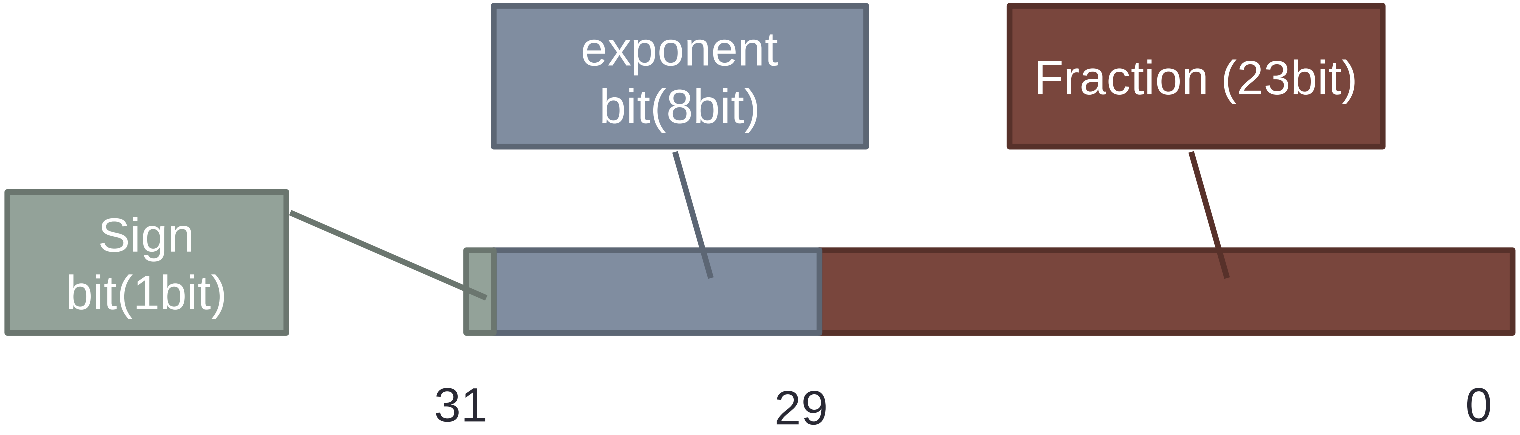
```
/* character data type initialization using character and ASCC*/
#include <stdio.h>

int main(void)
{
    char code1 = 'A';   // initialized code 1 using character constant
    char code2 = 65;    // initialized code 2 using ASCII code

    printf(“character constant= %c\n", code1);
    printf(“ASCII 65 is %c\n", code2);
   return 0;
}
```

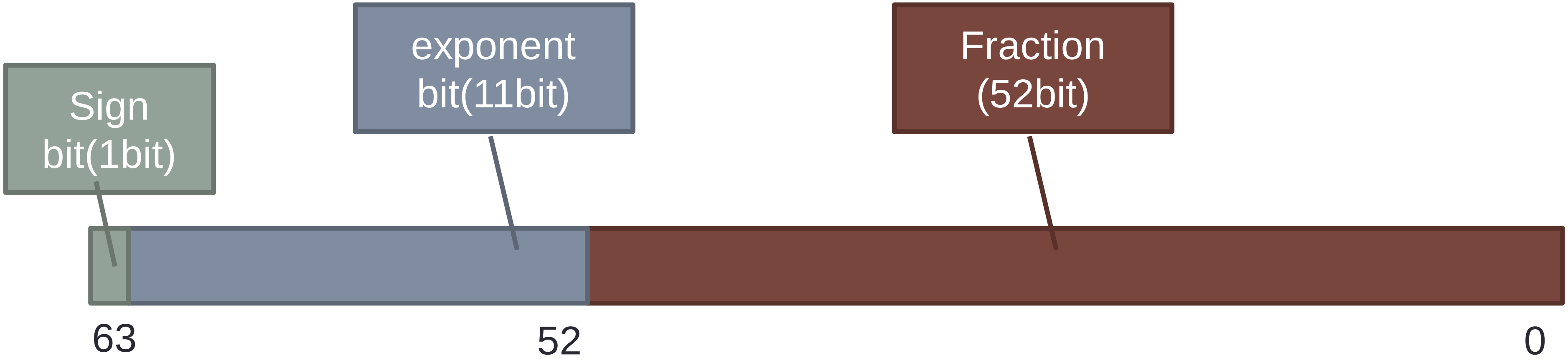
Floating point data type

- Float
  - ✓Seven significant digits



Floating point data type

- double
  - ✓15 significant digits



Example 6-1: significant digits

```
#include <stdio.h>

int main(void)
{
        float x = 1.2345678901234567890;
        double y = 1.2345678901234567890;

        printf(“size of float=%d\n", sizeof(float));
        printf(“size of double=%d\n", sizeof(double));

        printf("x = %30.25f\n",x);
        printf("y = %30.25f\n",y);
       return 0;
}
```

Example 6-2: scientific notation

- scientific notation
  - ✓1.23456e4 = 12345.6 = 1.23456×10^4
  - ✓1.23456e-3 = 0.00123456

```
#include <stdio.h>

int main(void)
{
        float y = 6.5e2;

        printf(“y= %f\n", y);
        printf(“y= %e\n", y);
return 0;
}
```

Exercise 7:overflow problem

- What is the expected results?
- Compare between expected results and execution results

```
#include <stdio.h>
int main(void)
{
        char x,x1,x2,x3;
        unsigned char y;
       x = -128;
       x1 = x-1;
       x2 = x-2;
       x3 = x-3;
        printf("x = %d\n",x);
        printf("x-1 = %d\n",x1);
        printf("x-2 = %d\n",x2);
        printf("x-3 = %d\n",x3);

        y = 256;
        printf("y = %u\n",y);   // unsigned data type using %u
        printf("y+1 = %u\n",y+1);
        printf("y+2 = %u\n",y+2);
        printf("y+3 = %u\n",y+3);
       return 0;
}
```

HW#2

- Complete the data type for the following table

Data type	Variable name	Initial value
	Grade	‘A’
	Weight	78kg
	Salary	2,000,000원
	Distance1	149,600,000km
	Price_of_apt	2,200,000,000원
	Height	178.9cm
	Distance2	2x10^{19}km
	Distance3	3x10^{123}km

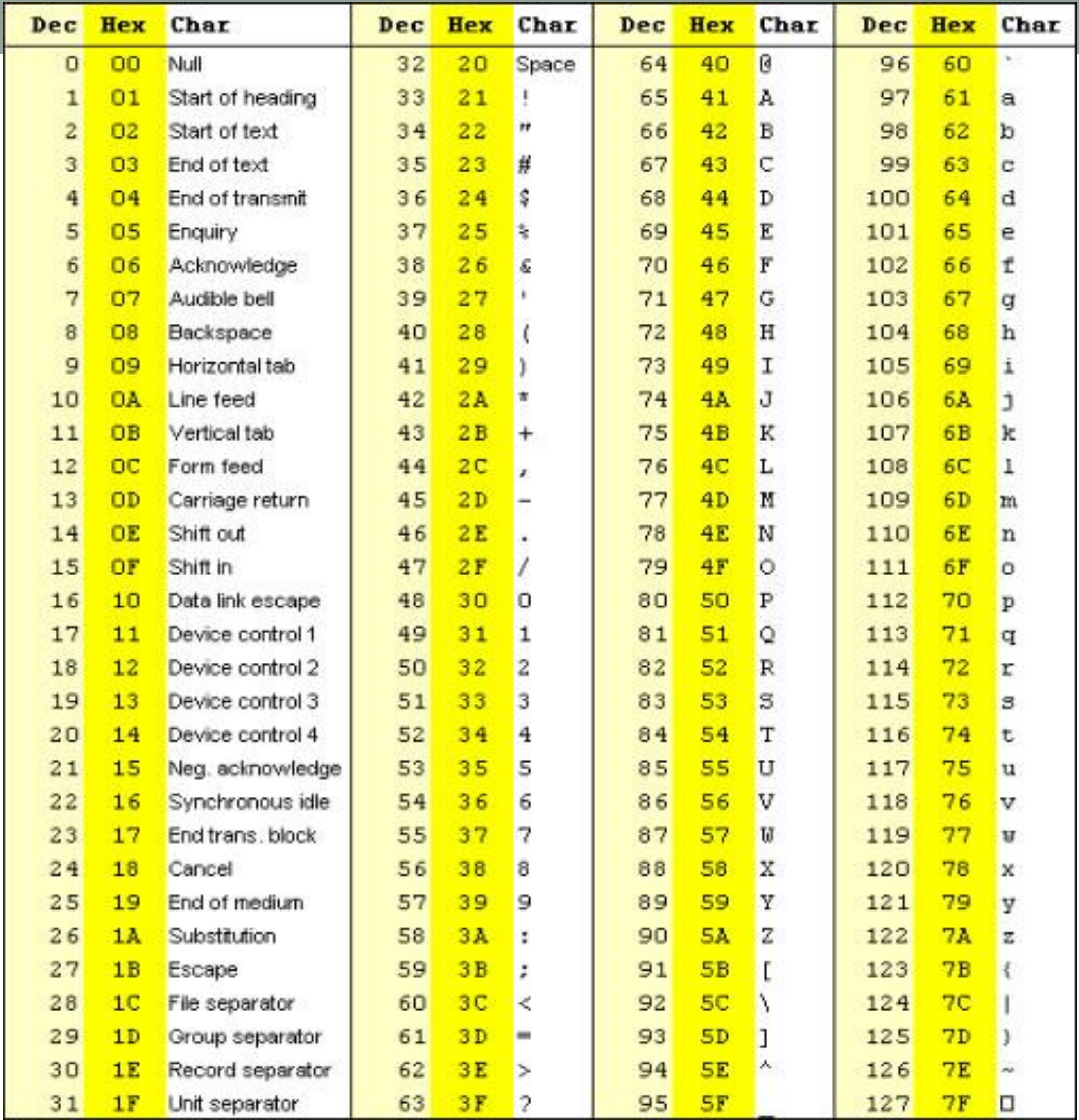
HW#2

- Complete the following source code using above table information **(refer the comment in the source)**
- **Execute the program and capture the result**
- **Convert from C to C++.**

```
#include <stdio.h>
int main()
{
_________________________; // declaration and initialization for variable grade
_________________________; // declaration and initialization for variable weight
_________________________; // declaration and initialization for variable salary
_________________________; // declaration and initialization for variable distance1
_________________________; // declaration and initialization for variable price_of_apt
_________________________; // declaration and initialization for variable height
_________________________; // declaration and initialization for variable distance2
_________________________; // declaration and initialization for variable distance3

printf("_______________________"); // print the variable grade using ASCII code
printf("_______________________"); // print the variable weight
printf("_______________________"); // print the variable salary
printf("_______________________"); // print the variable distance1
printf("_______________________"); // print the variable price_of_apt
printf("_______________________"); // print the variable height
printf("_______________________"); // print the variable distance2
printf("_______________________"); // print the variable distance3
return 0;
}
```

ASCII CODE



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□