C++ 基础（1）

一、目录

二、基础

1、数据类型

1、sizeof

查看占用内存大小

```
int a=10;
cout<<sizeof(a)<<endl;
```

2、char

单个字符（使用单引号引用）（只能放单个字符）

~~不可用双引号~~

```
char a='a';
cout<<a<<"\t"<<sizeof(a)<<endl;
cout<<(int)a<<endl;
```

ASCII码表格

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
0	NUL	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DCI	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	—	127	DEL

3、转义字符

转移字符	含义
换行	\n
反斜杠	\\
水平制表符(整齐输出数据)	\t
警示音	\a
回车	\r

4、字符串型

1、C语言

```
char a[]="hello world!";//双引号（数组形式）
cout<<a<<endl;
```

2、C++

`string 变量名 = "......"`

```
string a="hello world!";
cout<<a<<endl;
```

5、运算符

1、算术运算符

- ++a /--a `先自身++/--，再表达式运算`

  ```
  int a=10;
  int b=a++ * 10; //a=11  b=100
  ```

- a++ / a-- `先表达式运算，再自身++/--`

  ```
  int a=10;
  int b=++a * 10; //a=11  b=110
  ```

2、赋值运算符

运算符	术语	示例	结果
=	赋值	a=2;b=3;	a=2;b=3;
+=	加等于	a=0;a+=2;	a=2;
-=	减等于	a=5;a-=3;	a=2;
*=	乘等于	a=2;a*=2	a=4;
/=	除等于	a=4;a/=2;	a=2
%=	模等于	a=3;a%=2;	a=1;

3、三目运算符

a和b 比较 谁大返回谁

```
int a=10,b=20,c;
c = a > b ? a:b;
cout<<c<<endl; // c=20
```

2、程序流程

1、goto语句

跳转语句

```
int a=10,b=20;
goto FLAG;
a=30;
FLAG:
cout<<a<<endl; //a=10
```

3、数组

内存中连续内存空间

数组名也是首地址

1、一维数组

1、定义

```
int a[10];
int b[10]={0,0};
int c[]={1,1};
```

2、内存空间

```
int ing[10];
cout<<sizeof(ing)<<endl;
```

3、数组长度

```
int ing[10];
cout<<sizeof(ing)/sizeof(ing[0])<<endl;
```

4、首地址

```
cout<<ing<<endl;
cout<<(long long)ing<<endl;
```

2、二维数组

1、定义

```
int a[10][10];
int b[10][10]={{0,0},{1,1,1}};
int c[10][10]={0,0,0};
int d[][10]={0,1,2,3};
```

2、内存空间

```
cout<<a<<"  "<< sizeof(a)<<endl;
```

4、函数

1、声明

```
//函数声明
int compere(int a,int b);
//函数定义
int compere(int a,int b)
{
    return a > b ? a : b;
}
```

`声明可多次，定义就一次`

5、指针

`指针间接访问内存`

`通过指针保存一个地址`

1、定义

指针数据类型与原变量类型一致

```
int a=10;
int *p;
p=&a;
```

1、地址

`&a` `p`

2、值

`*p`

3、内存

指针大小为4字节（32位）

8字节（64位）

2、空指针

1、初始化

```
int *p=NULL;
```

2、访问权限

无访问权限，越界访问

3、野指针

```
int *p=(int *)0x1100;
```

4、*const*修饰指针

1、常量指针

> 指针指向可以改，指向的值不可以改
>
> const int *p=&a;

```
int main() {
    int a=10;
    const int *p=&a;
    *p=20;
    re
}
```

Read-only variable is not assignable

Change type of local variable 'p' to 'int *' Alt+Shift+Enter More actions... Alt+Enter

```
const int *p = &a
```

2、指针常量

> 指针的指向不可以改，指向的值可以改
>
> int * const p=&a;

```
int main() {
    int a=10,b=30;
    int * const p=&a;
    p=&b;
    r
}
```

Cannot assign to variable 'p' with const-qualified type 'int *const'

variable 'p' declared const here

Change type of local variable 'p' to 'int *' Alt+Shift+Enter More actions... Alt+Enter

```
int * const p = &a
```

3、即修饰指针也修饰常量

> 指针的指向不可以改，指向的值也不可以改
>
> const int * const p =&a;

```
int main() {
    int a=10,b=30;
    const int * const p=&a;
    *p=30;
    p=&b;
    return 0;
}
```

5、指针和数组

指针指向数组首地址，通过++实现数组往后移

1、定义

```
int arr[10]={1,2,3};
int *p=arr;
cout<<++*p<<endl;// 2
```

2、动态数组

```
int *a=new int[10];//申请空间
*a=10;
a[1]=20;
cout<<a[0]<<endl;
delete []a;//释放空间
```

6、指针和函数

```
void swap(int *p1,int *p2)
{
    int temp=*p1;
    *p1=*p2;
    *p2=temp;
}

int main() {
    int a=10,b=20;
    swap( p1: &a, p2: &b);
    cout<<a<<"\t"<<endl;
    return 0;
}
```

6、结构体

1、定义

一些类型组合成的一个类型

> 结构体
>
> C++中可以省略***struct***关键字
>
> 定义：

```
struct Student
{
 int m_age;
 string m_name;
 double m_score;
};
```

2、创建

- struct Student s1;

> s1.m_name
>
> s1.m_age;

- struct Student s2 = {18,"aaa",23.5};

> 赋初值

- 在定义结构体时就创建结构体变量

```
struct Student
{
    int m_age;
    string m_name;
    double m_score;
}s3;

int main() {
    s3.m_name="haizei";
    cout<<s3.m_name<<endl;
    return 0;
}
```

3、结构体数组

```
struct Student
{
    int m_age;
    string m_name;
    double m_score;
};

int main() {
    struct Student student[5]
    {
        {18,"aaa",98.5},
        {19,"bbb",98.5},
        {20,"ccc",98.5}
    };
    for(int i=0;i<sizeof(student)/ sizeof(student[0]);i++)
    {
        cout<<student[i].m_name<<endl;
    }
    return 0;
}
```

4、结构体指针

> 利用操作符`->`访问内容

```
Student s1={21,"小王",98.66};
Student *p=&s1;
cout<<p->m_name<<endl;
```

5、结构体嵌套结构体

```
struct Student
{
    int m_age;
    string m_name;
};
struct Teacher
{
    int m_age;
    string m_name;
    double m_score;
    struct Student student;
};
int main() {
    Teacher teacher={21,"maxin",98.0,{12,"gzy"}};
    cout<<teacher.student.m_name<<"\t"<<teacher.m_name<<endl;
    return 0;
}
```

6、结构体做函数参数

```
void Printf_1(struct Teacher teacher)//值传递
{
    cout<<teacher.student.m_name<<"\t"<<teacher.m_name<<endl;
};
void Printf_2(struct Teacher *teacher)//地址传递
{
    cout<<teacher->student.m_name<<"\t"<<teacher->m_name<<endl;
};

int main() {
    Teacher teacher={21,"maxin",98.0,{12,"gzy"}};
    Printf_1(teacher);
    Printf_2(&teacher);
    return 0;
}
```

7、结构体中 *const* 使用

```
void Printf_2(const Teacher *teacher)//地址传递
{
    cout<<teacher->student.m_name<<"\t"<<teacher->m_name<<endl;
    teacher->m_name="haiZei";
};
```

> 减少拷贝，减少内存开销，只读不可以修改