structure	**type**	**structure**	**type**	**type**
pointer	*	constant	void	
pointer array	&	variable	num	
reference		array	bool	
reference array			char	
		struct	struct_type	struct
		object	class	class
		enum	enum_type	enum

⑴**type**

1.struct_type

definition:struct struct_type_name{struct_body:data}

call:struct_type_name

2.class

definition:class class_name{class_body:data-attribute behavior-method}

class subclass_name:derived_way base_class{subclass_body:data-attribute behavior-method}

call:class_name

3.enum_type

definition:enum enum_type_name{enumerator=value}

call:enum_type_name

⑵**basic**

1.constant

definition:const data_type const_name=value

call:const_name

2.variable

definition:data_type var_name=value

call:var_name

3.array

definition:data_type array_name[index/key]={values}

call:array_name[index/key]

4.struct

definition:struct_type_name struct_name={arguments}

call:struct_name.member_name struct_name->member_name

5.object

definition:class_name object_name(constructor_arguments)

call:object_name.member_name object_name->member_name

6.struct array

definition:struct_type_name array_name[index/key]={arguments}

call:array_name[index/key].member_name array_name[index/key]->member_name

7.object array

definition:class_name array_name[index/key]={constructor_arguments}

call:array_name[index/key].member_name array_name[index/key]->member_name

8.enum

definition:enum_type_name enum_name

call:enum_name=enumerator~value

⑶advance

1.pointer

definition:prefix_type * pointer_name=address

call:pointer_name *pointer_name

instance:constant pointer variable pointer array pointer string pointer struct pointer object pointer

2.pointer array

definition:prefix_type * array_name[index/key]={address}

call---array_name[index/key] *array_name[index/key]

3.reference

definition:prefix_type & reference_name=target

call:reference_name

instance:constant reference variable reference array reference struct reference object reference

⑷function

1.function

definition:return_type function_name(input_type parameters){function_body}

call:function_name(arguments)

2.function pointer

definition:return_type * pointer_name(input_type parameters)=function_name

call:*pointer_name(arguments)

3.function pointer array

definition:return_type * pointer_name[index/key](input_type

parameters)={function_names}

call:*pointer_name[index/key](arguments)

⑸template

1.function template

definition:template <class parameters> parameters_type

function_name(parameters_type parameters_parameters)

{parameters_function_body}

call:function_name<arguments>

2.class template

definition:template <class parameters> class class_name{parameters_class_body}

call:class_name<arguments>