1. This is because C programmers need to manipulate memory, such as new; This requires the C programmer to know how much memory the type or variable is occupied.(the function of sizeof()).
2. Declaration doesn't allocate memory,but definition does.

Example:

Declaration:Max(int a,int b)

Definition:Max(int a,int b)

```
{
return a>b?a:b;
}
```

3. it can simplify our code;it can infer the types of variables according to the variables flowing.
4. (1) The reference must be initialized at the same time as it is created (pointers can be initialized at any time).

(2) Cannot have a NULL reference, the reference must be associated with a valid storage unit (pointers can be NULL).

(3) Once a reference is initialized, the relationship of the reference cannot be changed (pointers can change the object they refer to at any time).

Example:

```
Int a=10;
Int b=20;
Int c=&a;
Int *p;
P=&a;
P=&b;
```

5.(1)malloc and free are functions, and new and delete are operators

(2) The space applied for malloc will not be initialized, but new can be initialized

(3) When malloc applies for space, you need to manually calculate the space size and pass it, and new only needs to be followed by the space type.

(4)The return value of malloc is void*, which must be forcibly turned when used, and new is not needed, because new is followed by the type of space.

(5)When malloc fails to apply for space, NULL is returned, so it must be blank when used, new is not needed, but new needs to catch the exception.

(6) When applying for a custom type object, malloc/free will only open up the space and will not call the constructor and destructor, while new will call the constructor to complete the initialization of the object after applying for the space, and delete will call the destructor to clean up the resources in the space before releasing the space.

6.

```
#include <iostream>
#include <memory>
using namespace std;
int main()
{
    int i=10;
    auto_ptr<int>ap1(new int(4)),ap2;
    ap2=ap1;
```

```
	cout << *ap2;
	cout << *ap1 << endl;
	char *c;
	shared_ptr<char> sc(new char(10));
	c=sc.get();
	return 0;
}
```

7.

1)

a=12

b=3

c=45

d=634

2)

12345678901234567890

14 12 18

24 22 30

20 18 24

3)

2356

8.

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
	cout<<setw(5)<<"*"<<endl;
	cout<<setw(6)<<"***"<<endl;
	cout<<setw(7)<<"*****"<<endl;
	cout<<setw(8)<<"*******"<<endl;
	cout<<"*********";
}
```