Reference Variable

Syntax

<data type> & <ref_var> = var_name;

Ex

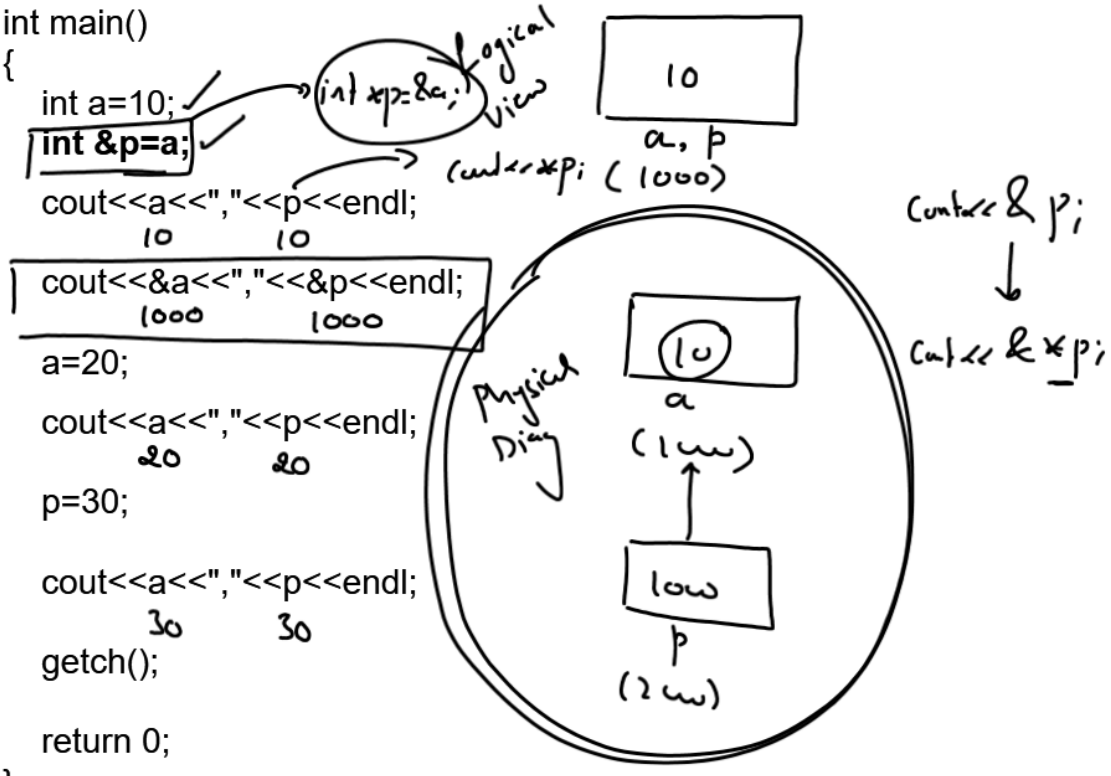
```
int a=10;
int &p=a;
```

```
char ch='z';
char &q=ch;
```

```
#include<iostream.h>
#include<conio.h>
int main()
{
   int a=10;
   int &p=a;
   cout<<a<<","<<p<<endl;
         10      10
   cout<<&a<<","<<&p<<endl;
         1000      1000
   a=20;
   cout<<a<<","<<p<<endl;
         20      20
   p=30;

   cout<<a<<","<<p<<endl;
         30      30
   getch();

   return 0;
}
```



```
char ch = 'A';
```

What is a reference variable?
==========

1. Reference variables are a **new mechanism** introduced by C++ and not present in C lan.

2. Using a reference variable a programmer can refer or access a single memory location using another **alternate name.**

3. For a programmer , reference variable is simplified version of pointer.

4. That is they are as powerful as pointer but have an easy syntax compared to pointers.

5. Just like a pointer operates on other variable , similarly a reference variable also operates on other variables.

6. But to access a variable's value using pointer we have to use indirection operator i.e. we have to dereference the pointer while using reference variable no indirection operator is required

Some important points about reference variable
==========
Q 1. Does a reference variable share the same memory location or address as of the variable to which it is refering?

Ans 1. No, a BIG NO

Although when we print the address of a reference variable we always get the same address as that of the variable to whom the reference variable is refering and due to this we might thing that referece variable share the address of the same variable

But this is not at all true.

In reality a reference variable is actually a pointer and so it has totally different address compared to address of the variable.

But the C++ compiler never reveals the real address of reference variable to the programmer because compiler wants us to believe that reference variable are aliases to other variable. Thus whenever we use the name of a reference variable in our program the compiler simply prefixes it with indirection operator.

So
cout<<p;
becomes
cout<<*p;
and
cout<<&p;
becomes
cout<<&(*p);