- Parameters

  - *Value* parameters: the function receives a copy of the **value** of the actual parameter (*pass by value*)

  - Changing the values of the formal parameters **does not** affect the values of the actual parameters !!

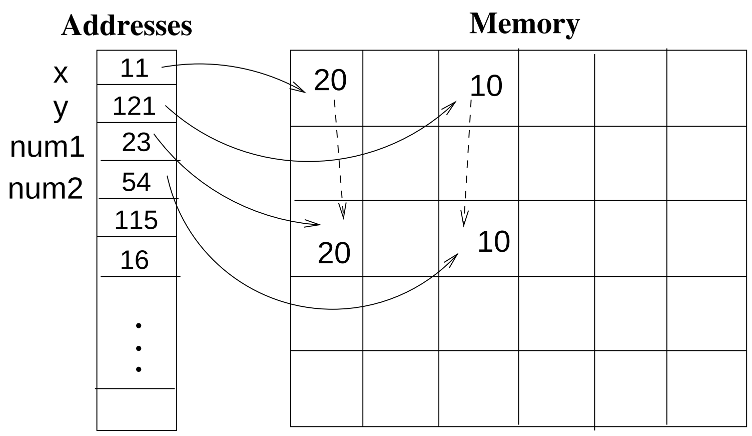
  - *Reference* parameters: the function receives a copy of the **address** of the actual parameter (*pass by reference*)

  - When the address of a parameter is passed to a function, an **ampersand (&)** must be attached at the end of the parameter's datatype

  - Changing the values of the formal parameters **does** affect the values of the actual parameters !!

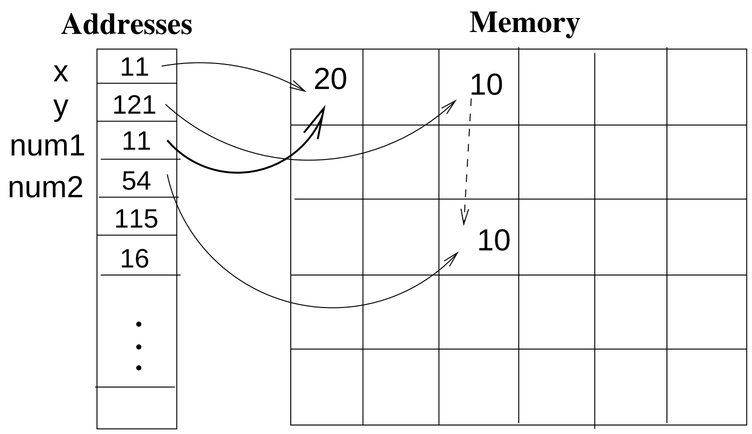
Value parameters

```
void Test(int , int);
void main()
{
 int x,y;
     ...
  Test(x,y);
    ...
}

void Test(int num1, int num2)
{
   ...
}
```



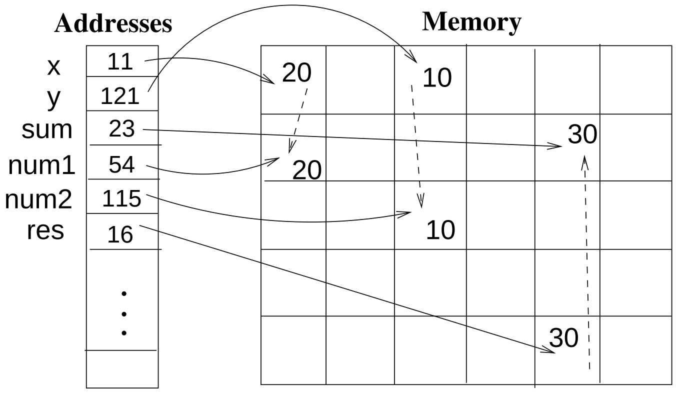
Reference parameters

```
void Test(int& , int);
void main()
{
 int x,y;
     ...
  Test(x,y);
    ...
}

void Test(int& num1, int num2)
{
   ...
}
```



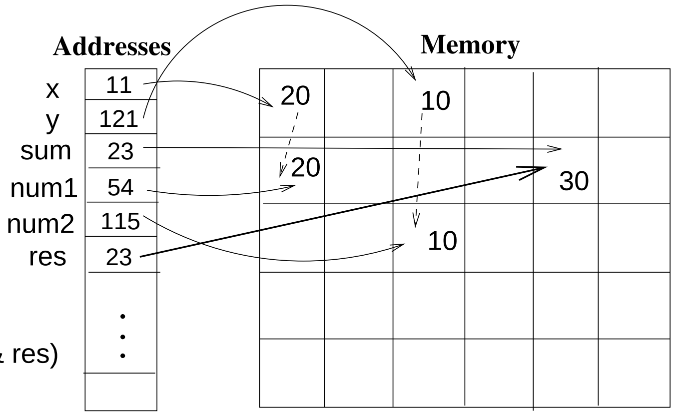
compute_sum() using "pass by value"

```
#include <iostream.h>
int compute_sum(int , int);

void main()
{
 int x,y,sum;
    ...
 sum=compute_sum(x,y);
 cout << sum;
    ...
}

int compute_sum(int num1, int num2)
{
 int res;

 res=num1 + num2;
 return res;
}
```



compute_sum() using "pass by reference"

```
#include <iostream.h>
void compute_sum(int , int, int&);

void main()
{
 int x,y,sum;
    ...
 compute_sum(x,y,sum);
 cout << sum;
    ...
}

void compute_sum(int num1, int num2, int& res)

{
 res=num1 + num2;
}
```



- In *pass by value*, any expression can be passed to a function but in *pass by reference*, only the address of a variable can be passed to a function

```
void compute_sum(int,int,int&)
void main()
{
  int x,y,sum;

  x=10; y=20;
  compute_sum(x,y,sum);
  compute_sum(2*x-1, y-15, sum);
}

void compute_sum(int num1, int num2, int& res)
{
 res = num1 + num2;
}
```