- Matching the data type of the actual and formal parameters

  - The case of "value" variables

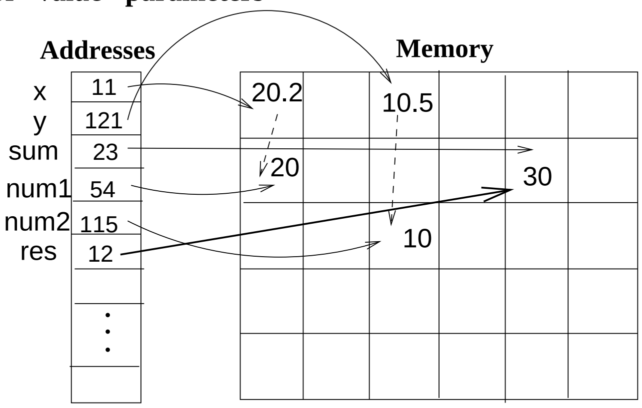
    * The data type of the actual parameters is coerced to the data type of the formal parameters

  - The case of "reference" variables

    * The value of the actual parameter is copied to a temporary variable of the correct type
    * The address of the temporary variable is passed to the function
    * The value of the actual parameter does not change !!

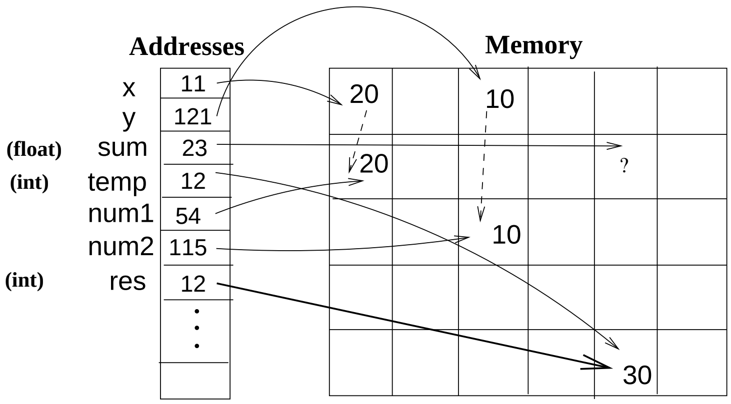
The case of "value" parameters

```
#include <iostream.h>
void compute_sum(int , int, int&);
void main()
{
 float x,y;
 int sum;
     ...
 compute_sum(x,y,sum);
 cout << sum;
     ...
}

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



The case of "reference" parameters

```
#include <iostream.h>
void compute_sum(int , int, int&);
void main()
{
 int x,y;
 float sum;
     ...
 compute_sum(x,y,sum);
 cout << sum;
     ...
}

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



- Examples

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

void main()
{
 int i, j;

 i=1;
 j=2;

 change(i,j);
 cout << i << j;
}

void change(int& x, int y)
{
 x=3;
 y=4;
}


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

void main()
{
 int i,j,k;

 i=1;
 j=2;
 k=3;

 scramble(i,j,k);
 cout << i << j << k;
}

void scramble(int x, int& y, int z)
{
 x=10;
 y=20;
 z=30;
}
```

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

void main()
{
 int i,j,k;

 i=1;
 j=2;
 k=3;

 scramble(j,j,j);
 cout << i << j << k;
}
```

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

void main()
{
 int i,j,k;

 i=1;
 j=2;
 k=3;

 scramble(k,j,i);
 cout << i << j << k;
}

void scramble(int x, int& y, int& z)
{
 x=10;
 y=20;
 z=30;
}
```

Recommended exercises

- *Quick Check* excersices: 2-7

- *Exam preparation* excersices: 1-4, 7-14