

PIC 10A 2020 W



Surface $x^{\frac{2}{3}} + y^{\frac{2}{3}} + z^{\frac{2}{3}} = 1$

Normal vector to the surface

$$\left\langle \frac{2}{3}x_0^{-\frac{1}{3}}, \frac{2}{3}y_0^{-\frac{1}{3}}, \frac{2}{3}z_0^{-\frac{1}{3}} \right\rangle$$

Planes tangent to the surface

$$\frac{2}{3}x_0^{-\frac{1}{3}}(x-x_0) + \frac{2}{3}y_0^{-\frac{1}{3}}(y-y_0) + \frac{2}{3}z_0^{-\frac{1}{3}}(z-z_0) = 0$$

$$\Rightarrow \underbrace{\frac{2}{3}x_0^{-\frac{1}{3}}x}_{x_0^{-\frac{1}{3}}x} - \underbrace{\frac{2}{3}x_0^{\frac{2}{3}}}_{x_0^{\frac{2}{3}}} + \underbrace{\frac{2}{3}y_0^{-\frac{1}{3}}y}_{y_0^{-\frac{1}{3}}y} - \underbrace{\frac{2}{3}y_0^{\frac{2}{3}}}_{y_0^{\frac{2}{3}}} + \underbrace{\frac{2}{3}z_0^{-\frac{1}{3}}z}_{z_0^{-\frac{1}{3}}z} - \underbrace{\frac{2}{3}z_0^{\frac{2}{3}}}_{z_0^{\frac{2}{3}}} = 0$$

$$x_0^{-\frac{1}{3}}x + y_0^{-\frac{1}{3}}y + z_0^{-\frac{1}{3}}z = x_0^{\frac{2}{3}} + y_0^{\frac{2}{3}} + z_0^{\frac{2}{3}} = 1$$



$$x_0^{-\frac{1}{3}}x + y_0^{-\frac{1}{3}}y + z_0^{-\frac{1}{3}}z = 1$$

x -intercept: $y=0$ $z=0$

$$x_0^{-\frac{1}{3}}x = 1 \Rightarrow x = x_0^{\frac{1}{3}}$$

y -intercept: $x=0$ $z=0$

$$y_0^{-\frac{1}{3}}y = 1 \Rightarrow y = y_0^{\frac{1}{3}}$$

z -intercept: $x=0$ $y=0$

$$z_0^{-\frac{1}{3}}z = 1 \Rightarrow z = z_0^{\frac{1}{3}}$$

The sum of squares:

$$(x_0^{\frac{1}{3}})^2 + (y_0^{\frac{1}{3}})^2 + (z_0^{\frac{1}{3}})^2 = 1$$

$$!(a \parallel (b \& \& !a) \&& !c)$$

$$= !(a \parallel ((b \& \& !a) \&& !c))$$

$$= !(T \parallel ((T \& \& F) \&& F))$$

$$= !(T \parallel (F \& \& F))$$

$$= !T$$

$$!(a \parallel ((b \& \& !a) \&& !c))$$

$$= !a \&& !(b \& \& (!a) \&& !c)$$

$$= !a \&& (!b \parallel !a) \parallel c$$

$$= !a \&& ((!b \parallel a) \&& !c)$$

$$= F \&& ((F \parallel T) \&& F)$$

$$= F \&& (T \& \& F)$$

$$= F \&& F$$

$$= F$$

```

int a = 0;
int b = 0;
int f(int c)
{
    int n = 0;
    a = c;
    if (n < c)
        n = a + b;
    return n;
} n=1

```

$c=1$
 $n=0$
 ~~$a=1$~~

$n = a+b = 1$


```

int g(int c)
{
    int n = 0;
    int a = c;
    if (n < f(c))
        n = a + b;
    return n;
}

```

$c=1$
 $n=0$
 $a=1$
 $c=1 \rightarrow c=1$

$n = a+b = 1$


```

int main()
{
    int i = 1;
    int b = g(i);
    cout << a + b + i << "\n";
    return 0;
}

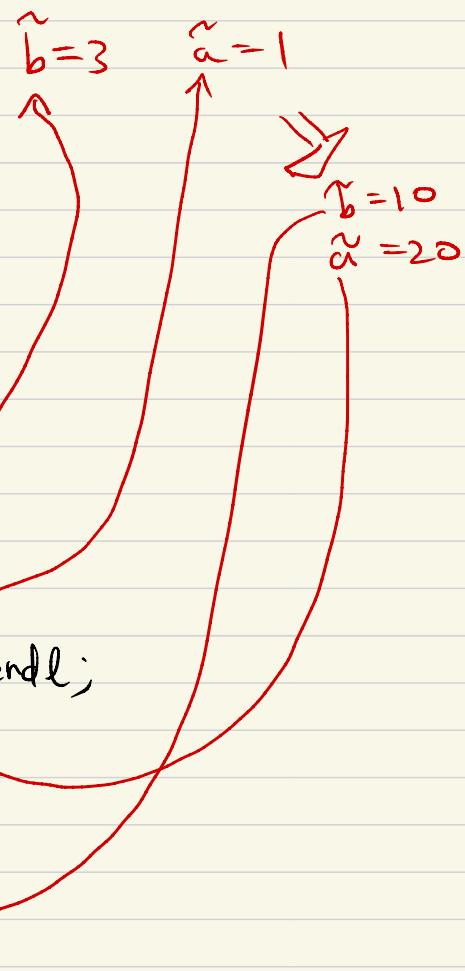
```

$c=1$
 $i=1$
 $a+b=1$
 $b=g(i)=1$
 \downarrow
 $b=1$

$a+b+i = 1+1+1=3$

```
void change(int& b, int& a)  
{  
    b = 10;  
    a = 20;  
}
```

```
int main()  
{  
    int a = 3;  
    int b = 1;  
    change(a, b);  
    cout << a << " " << b << endl;  
}
```



10 20

```
void fun( int & a)  
{  
    a *= 2;  
}  
  
int main()  
{  
    int b = 3;  
    fun(b);  
    fun(b);  
    cout << b << endl;  
}
```

The diagram illustrates the state of variables in memory:

- Variable a:** A pointer variable declared in the `fun` function. It points to a memory location labeled `a`. A red circle highlights the variable `a`, and a red arrow points from the label `a` to the variable's declaration.
- Variable b:** A local variable declared in the `main` function. It has a value of 3. A red circle highlights the variable `b`, and a red arrow points from the label `b` to the variable's declaration.
- Assignment:** In the first call to `fun(b)`, the value of `b` is copied into the parameter `a`. This is shown by a red arrow pointing from the variable `b` to the variable `a`.
- Second Call:** In the second call to `fun(b)`, the value of `b` is again copied into the parameter `a`. This is shown by another red arrow pointing from the variable `b` to the variable `a`.

```
void swap( int &a , int
```

Exercise P4.2. Write a procedure void sort2(int& a, int& b) that swaps the values of a and b if a is greater than b and otherwise leaves a and b unchanged. For example,

```
int u = 2;  
int v = 3;  
int w = 4;  
int x = 1;  
sort2(u, v); // u is still 2, v is still 3  
sort2(w, x); // w is now 1, x is now 4
```

void sort2(int& a, int & b)

{

// TODO
if (a > b)
{

```
    int temp = a;  
    a = b;  
    b = temp;
```

}

}

void sort3(int& a, int& b, int& c)

{

// TODO

sort2(b, c); $\rightarrow b=2 \quad c=1 \Rightarrow$ ~~b=2~~ ~~c=1~~
sort2(a, c); $\rightarrow a=3 \quad c=2 \Rightarrow$ ~~a=3~~ $c=3$
sort2(a, b); $\rightarrow a=1 \quad b=2 \Rightarrow$ $a=1$ $b=2$

}

int main()

{

```
int u = 3; int w = 1;  
int v = 2;
```

cout << u << v << w << endl; $\rightarrow 3 \ 2 \ 1$

sort3(u, v, w);

cout << u << v << w << endl; $\rightarrow 1 \ 2 \ 3$

$a=1 \quad b=2 \quad c=3$

```
void sort3( int& a , int& b, int& c)
```

```
{
```

```
    int tempa=a;  
    int tempb=b;  
    int tempc=c;
```

```
    if ( a > b && b > c)
```

```
{
```

```
    a=tempa;  
    b=tempb;  
    c=tempc;
```

```
}
```

```
    else if ( a > c && c > b)
```

```
{
```

```
    a=tempa;  
    b=tempc;  
    c=tempb;
```

```
}
```

```
    else if ( b > a && a > c)
```

```
{
```

```
    a=tempb;  
    b=tempa;  
    c=tempc;
```

```
bool leap-year (int year)
```

```
{
```

```
    return (year - 2000) % 4 == 0
```

```
}
```

$$\text{year} = 2000 \quad 0 \% 4 = 0 \Rightarrow \text{true}$$

$$\text{year} = 901 \quad (901 - 2000) \% 4 = -1099 \% 4 \neq 0$$

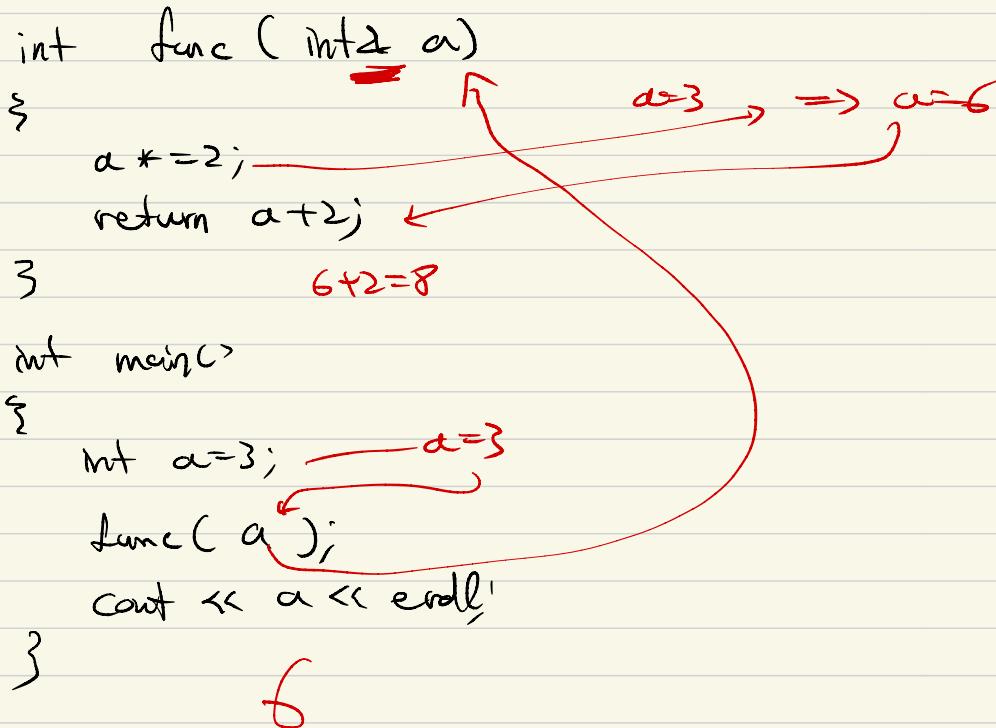
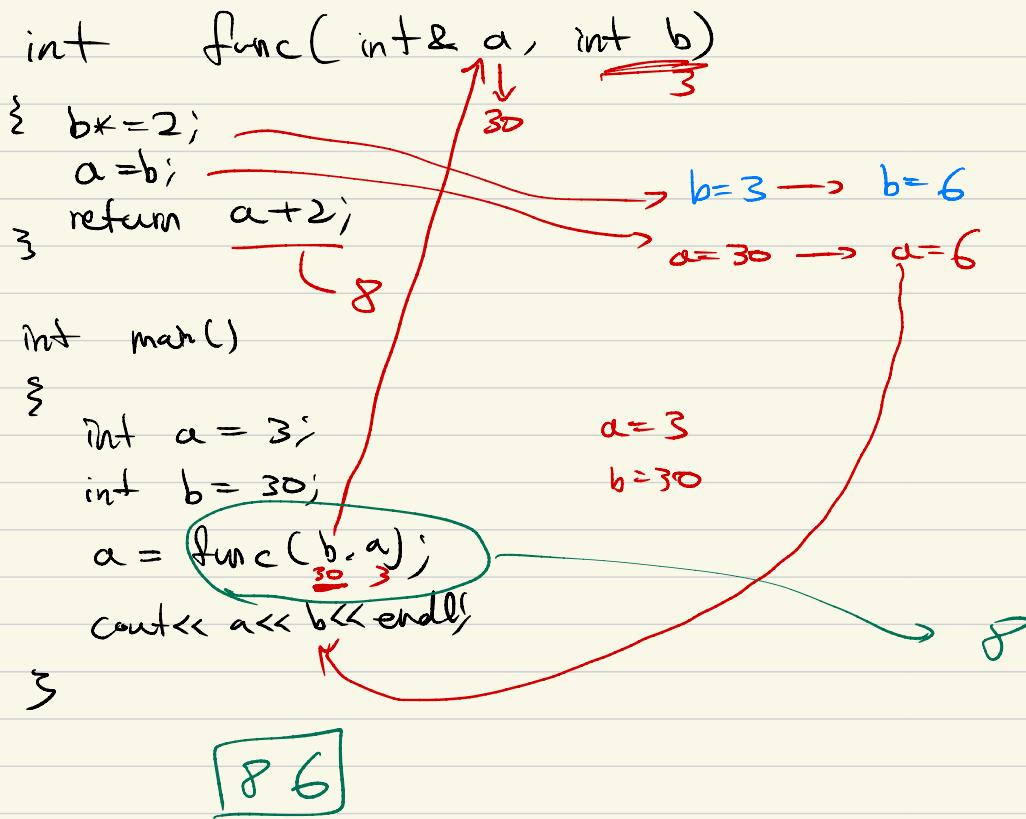
↓
false

```
int main()
```

```
{
```

```
cout << leap-year(2000) << endl; // 1
```

```
cout << leap-year(2009) << endl; // 0
```



Exercise P5.1. Implement all member functions of the following class:

```
class Person
{
public:
    Person(); // constructor
    Person(string pname, int page);
    void get_name() const; // member functions ) Accessor
    void get_age() const;
private:
    string name;
    int age; // 0 if unknown
};

Person::Person() // Default constructor
{
    name = "John";
    age = 0;
}

Person::Person(string pname, int page) // constructor
{
    name = pname;
    age = page;
}

void Person::get_name() const // accessor
{
    cout << "name : " << name << endl;
}

void Person::get_age() const
{
    cout << "age : " << age << endl;
}
```

```
#include <iostream>
#include <string>

using namespace std;
```

```
class PEmployee
```

```
{  
    PEmployee :: PEmployee()  
}
```

```
    name = "John";  
    salary = 0;
```

```
}  
PEmployee :: PEmployee(string employee_name,  
                      double initial_salary)
```

```
{  
    name = employee_name;  
    salary = initial_salary;
```

```
}  
double PEmployee :: get_salary() const  
{  
    return salary;
```

```
}  
string PEmployee :: get_name() const  
{  
    return name;
```

void PEmployee::set_salary(double new_salary)
{

salary = new_salary;

}

class Account

{

public :

Account();

Account(int pbalance);

)

constructors

void add(double money);

void withdraw(double money);

)

mutators

O = O

double get_money() const;) access, return (double)

private:

double balance;



3

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
vector<int> v1; {3}
```

Goal: $v1 = \{1, 2, \dots, 100\}$

```
for (int i = 1; i <= 100; ++i)
```

```
{
```

```
v1.push_back(i);
```

```
}
```

```
cout << v1.size() << endl; // prints out 100
```

```
for (int i = 0; i < v1.size(); ++i)
```

```
{
```

```
cout << v1[i] << endl;
```

```
}
```

$v1 = \{1, 2, 3, 4, 5, \dots, 100\}$

$v1[10]$ $v1[0]$

1
2
3
4
:
99
100

$$a = \{1, 2, 3\}$$

$$b = \{2, 0, 3\}$$

$$2 + 0 + 3 = 5$$

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

```
double fun (vector<double> a, vector<double> b)
```

```
{ double s=0;
```

```
for (int i=0; i<a.size(); ++i)
```

```
{
```

```
    s += a[i] * b[i];
```

```
    }
```

$$a[i] \quad b[i]$$

$$1 \times 0 = 0$$

$$2 \times 3 = 6$$

```
return s;
```

```
int main()
```

```
{
```

```
vector<double> v1;
```

```
vector<double> v2;
```

```
v1.push_back(1);
```

(1, 2)

```
v1.push_back(2);
```

(0, 3)

```
v2.push_back(0);
```

```
v2.push_back(3);
```

```
cout << fun(v1, v2) << endl;
```

```
}
```

get 6.

```
class Character {
```

```
public:
```

```
Character();
```

$$\text{Object} = \text{Object}$$

```
void Character::set_health(int newh)
```

```
}
```

```
health = newh;
```

```
},
```

```
private:
```

```
string name;  
int health;  
int damage;  
int arrows;
```

```
string Character::get_name() const
```

```
{
```

```
return name;
```

```
int Character::get_health() const
```

```
{
```

```
return health;
```

class Character

{
public:
 constructors
 mutators
 accessors
}
private: ← optional
}; ;
*

The diagram illustrates a C++ class definition. It starts with the keyword 'class' followed by the class name 'Character'. This is enclosed in a yellow circle. Below it is the opening brace '{'. To the right of the brace is the keyword 'public:' also in a yellow circle. Inside the class body, there is a box containing 'constructors', 'mutators', and 'accessors'. Below the class body is the keyword 'private:' followed by a note '← optional'. At the very bottom is the closing brace '}' followed by a semicolon ';' in a yellow circle. A red arrow points from the start of the class keyword to the opening brace. Another red arrow points from the end of the public section to the closing brace. Red asterisks are placed at the beginning and end of the entire class block.

```
#include <iostream>
#include <vector>
```

INSERT

```
void insert (vector<int>& v1, vector<int>& v2, int start)
```

```
{
```

TODO

}

```
int main()
```

```
{
```

```
vector<int> v1 = {0, 1, 2, 3, 4, 5};
```

```
vector<int> v2 = {-1, -2, -3};
```

insert(v1, v2, 2);

v1 = {0 1 2 3 4 5};

v1 = {0 1 -1 -2 -3 2 3 4 5};

insert(v1, v2, 0)

v1 = {-1 -2 -3 0 1 2 3 4 5};

insert(v1, v2, 1)

v1 = {0 -1 -2 1 2 3 4 5};

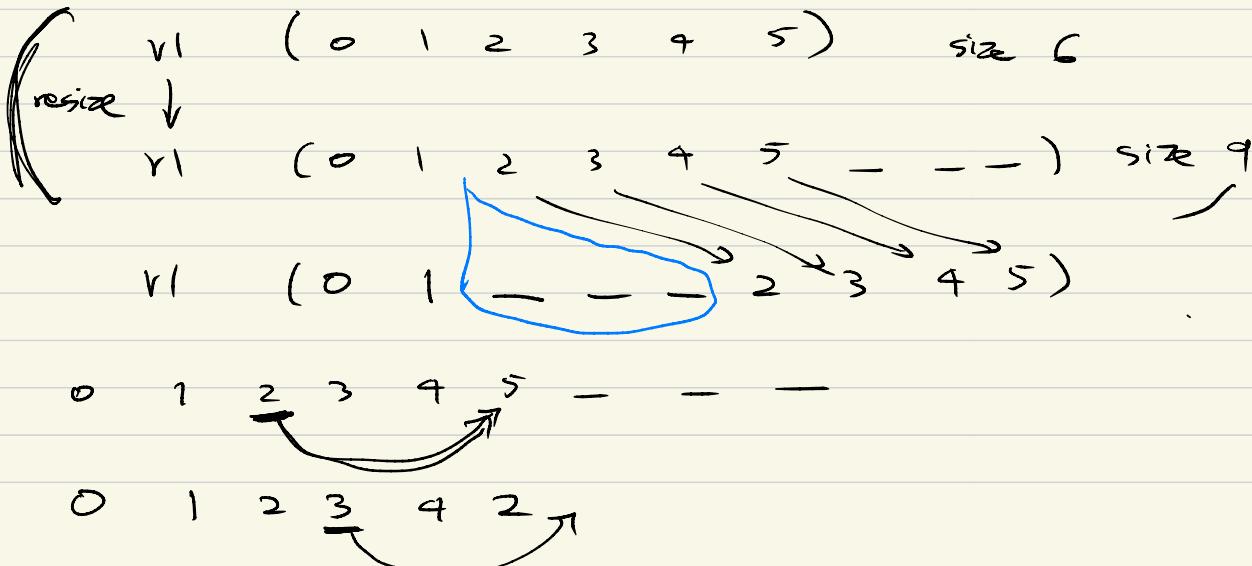
insert(v1, v2, 2)

v1 = {0 1 -1 -2 2 3 4 5};

:

insert(v1, v2, 6)

v1 = {0 1 2 3 4 5 -1 -2 -3};



0 1 2 3 4 2 3

0 1 2 3 4 2 3 4

0 1 2 3 4 2 3 4 2

0 1 2 3 + 5 - - -

0 1 2 3 4 5 - - 5

0 1 2 3 4 5 - 4 5

0 1 2 3 4 5 3 4 5

0 1 2 3 + 2 3 4 5

v1 = 0 1 2 3 4 5
v1.size() v2.size() v1.size() - 1

v1[8] = v1[5];

v1[7] = v1[4];

v1[6] = v1[3];

v1[5] = v1[2];

for(int i=5; i>=2; i--)
{
 v1[i+3] = v1[i];
}
3
v2.size()

0 1 2 3 4 2 3 4 5
 } we want

Recall

$$r2 = \{-1, -2, -3\}$$

0 1 -1 -2 -3 2 3 4 5

$$v1[2] = r2[0];$$

$$v1[3] = r2[1];$$

$$v1[4] = r2[2];$$

$\left\{ \begin{array}{l} \text{for (int } i=0; i < 3, ++i) \\ \quad v1[i-2] = r2[i]; \\ \end{array} \right.$

start

void insert (...)

{

int v1s = v1.size() // 6

int r2s = r2.size() // 3

resize ✓

v1.resize(v1s + r2s);

for (int i = v1s - 1; i >= start; --i)

move ✓

{
 $v1[i+r2s] = r1[i];$

}

replace ✓

for (int i = 0; i < r2s; ++i)

{

$v1[i+start] = r2[i];$

}

3

Remove

~~rl = {0, 1, 2, 3, 4, 5}~~

remove (rl, 0)

$\rightarrow rl = \{1, 2, 3, 4, 5\}$

remove (rl, 1)

$\rightarrow rl = \{0, 2, 3, 4, 5\}$

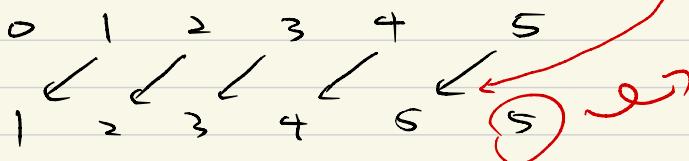
remove (rl, 2)

$\rightarrow rl = \{0, 1, 3, 4, 5\}$

remove (rl, 5)

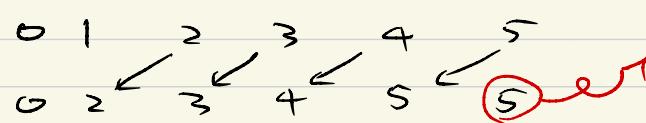
$\rightarrow rl = \{0, 1, 2, 3, 4\}$

remove (rl, 0)

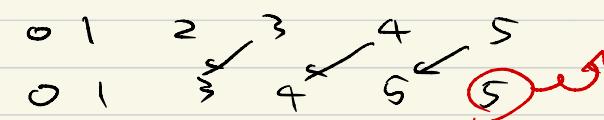


The for loop
always ends at
index $rl.size() - 1$

remove (rl, 1)



remove (rl, 2)



$rl[2] = rl[3];$
 $rl[3] = rl[4];$
 $rl[4] = rl[5];$

$\left(\begin{array}{l} \text{for (int } i=2; i < 5; ++i) \\ \{ \\ \quad rl[i] = rl[i+1]; \\ \end{array} \right)$

$rl.size() - 1$

void remove (vector<int> &rl, int start)
{

 int rls = rl.size();

 for (int i = start; i < rls - 1; ++i)

 rl[i] = rl[i + 1];

 rl.pop_back();

move ✓

remove the last ✓

 }