实验 JDBC编程和PL/pgSQL函数

主要内容

JDBC编程

- Java环境配置
- Java应用访问数据库

PL/pgSQL函数

- 程序的基本结构
- 基本语法
- 函数的编写

主要内容

JDBC编程

- Java环境配置
- Java应用访问数据库

PL/pgSQL函数

- 程序的基本结构
- 基本语法
- 函数的编写

Java环境配置

- Eclipse——开发平台
- jdk——java的开发工具集
- jre——java程序的运行环境

- **查看是否已安装好jdk和jre (java)**
  - **cmd　输入 java –version**

1. Eclipse安装

eclipse-win64.zip

- Eclipse官网下载（32位/64位）
- 解压缩即安装成功

名称	修改日期	类型	大小
configuration	2019/10/23 15:39	文件夹	
dropins	2017/4/12 15:29	文件夹	
features	2017/4/12 15:27	文件夹	
p2	2017/4/12 15:27	文件夹	
plugins	2017/4/12 15:27	文件夹	
readme	2017/4/12 15:27	文件夹	
.eclipseproduct	2015/1/28 10:08	ECLIPSEPRODUCT ...	1 KB
artifacts.xml	2015/2/19 3:31	XML 文档	249 KB
eclipse.exe	2015/2/19 3:32	应用程序	314 KB
eclipse.ini	2015/2/19 3:31	配置设置	1 KB
eclipsec.exe	2015/2/19 3:32	应用程序	26 KB
epl-v10.html	2015/1/28 10:08	Chrome HTML Doc...	13 KB
notice.html	2015/1/28 10:08	Chrome HTML Doc...	9 KB

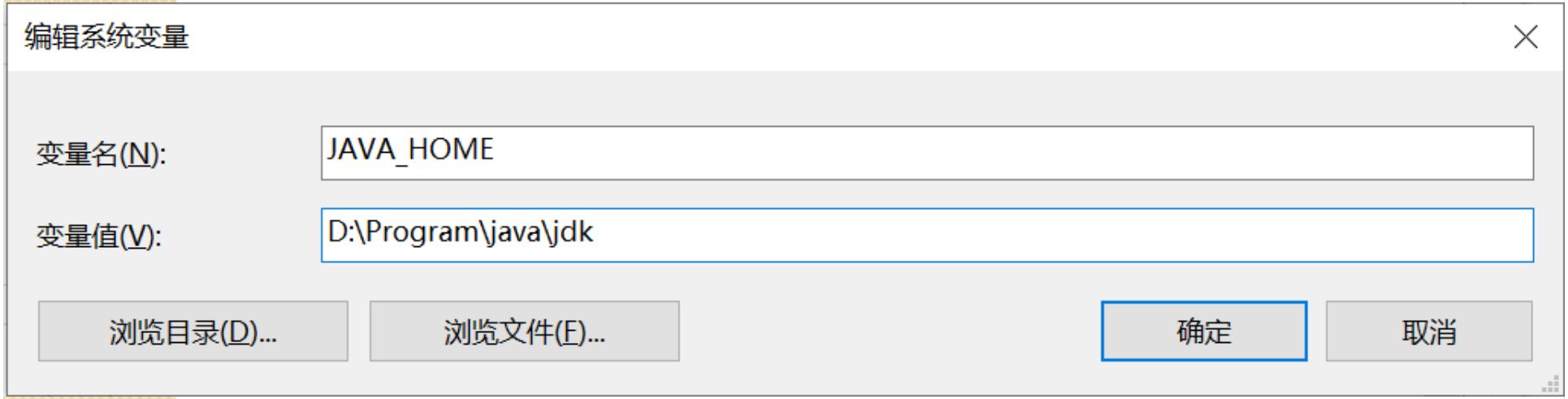
2. jdk和jre(java)的安装

- oracle官网下载jdk
  - 下载的jdk里面必然含有jre，安装上jdk就**不需要**再安装单独的jre
- 安装
  - 它会要求你选择**两次**安装路径，第一次是jdk，第二次是jre
  - 注意：jdk和jre要放在同一个安装目录下

2. jdk和jre(java)的安装

- 配置环境变量
  - 我的电脑，右键“属性”-“高级”-“环境变量”-“系统变量”
  - 系统变量 - 新建 - JAVA_HOME 变量

（变量值为jdk安装目录，注意最后不加分号）



2. jdk和jre(java)的安装

- 配置环境变量
  - 系统变量-找到Path变量-编辑

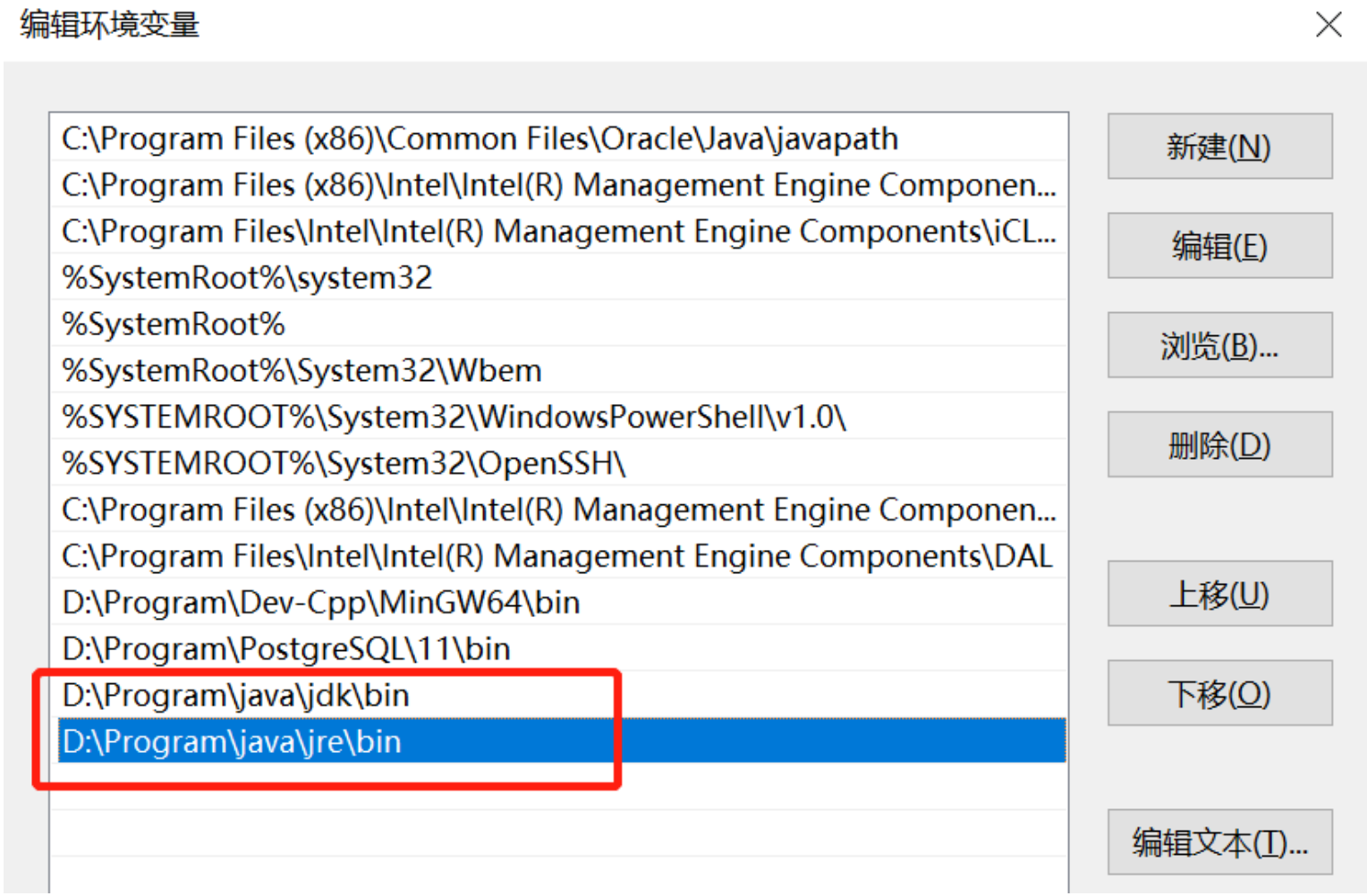
    在变量值最后输入(按照自己的安装目录更改):

    D:\Program\Java\jdk\bin;D:\Program\Java\jre\bin;

    （注意原来Path的变量值末尾有没有分号，如果没有，先输入分号再输入上面的代码）

2. jdk和jre(java)的安装

- 配置环境变量



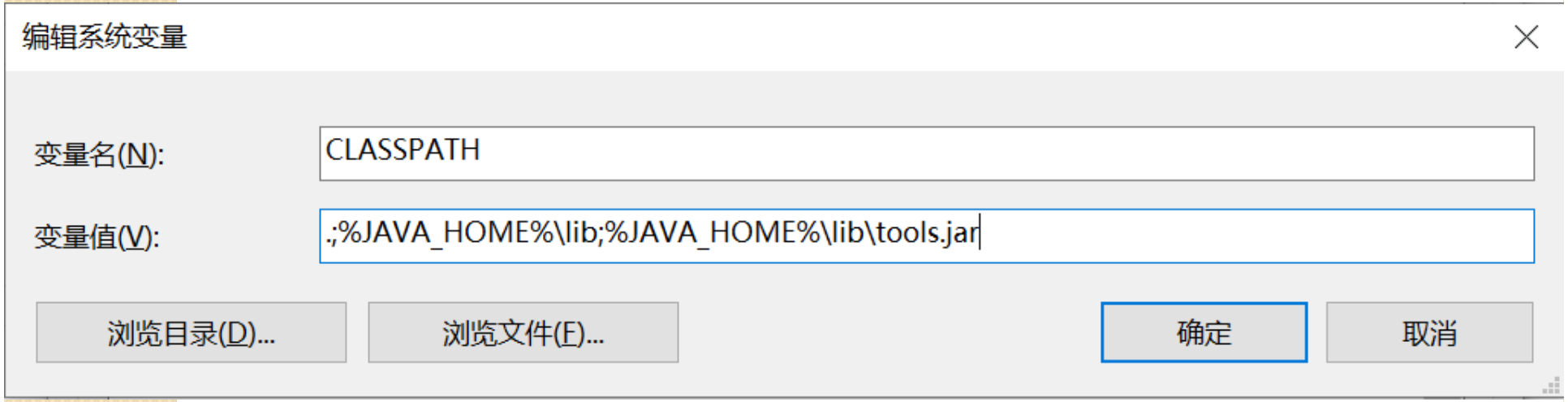
2. jdk和jre(java)的安装

- 配置环境变量
  - 系统变量 - 新建 CLASSPATH 变量

变量值填写：

.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar

（注意最前面有一点）



2. jdk和jre(java)的安装

- 配置环境变量
  - 检查是否已安装好jdk和jre (java)

    cmd　输入 java –version

```
C:\Users\24836>java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)

C:\Users\24836>
```

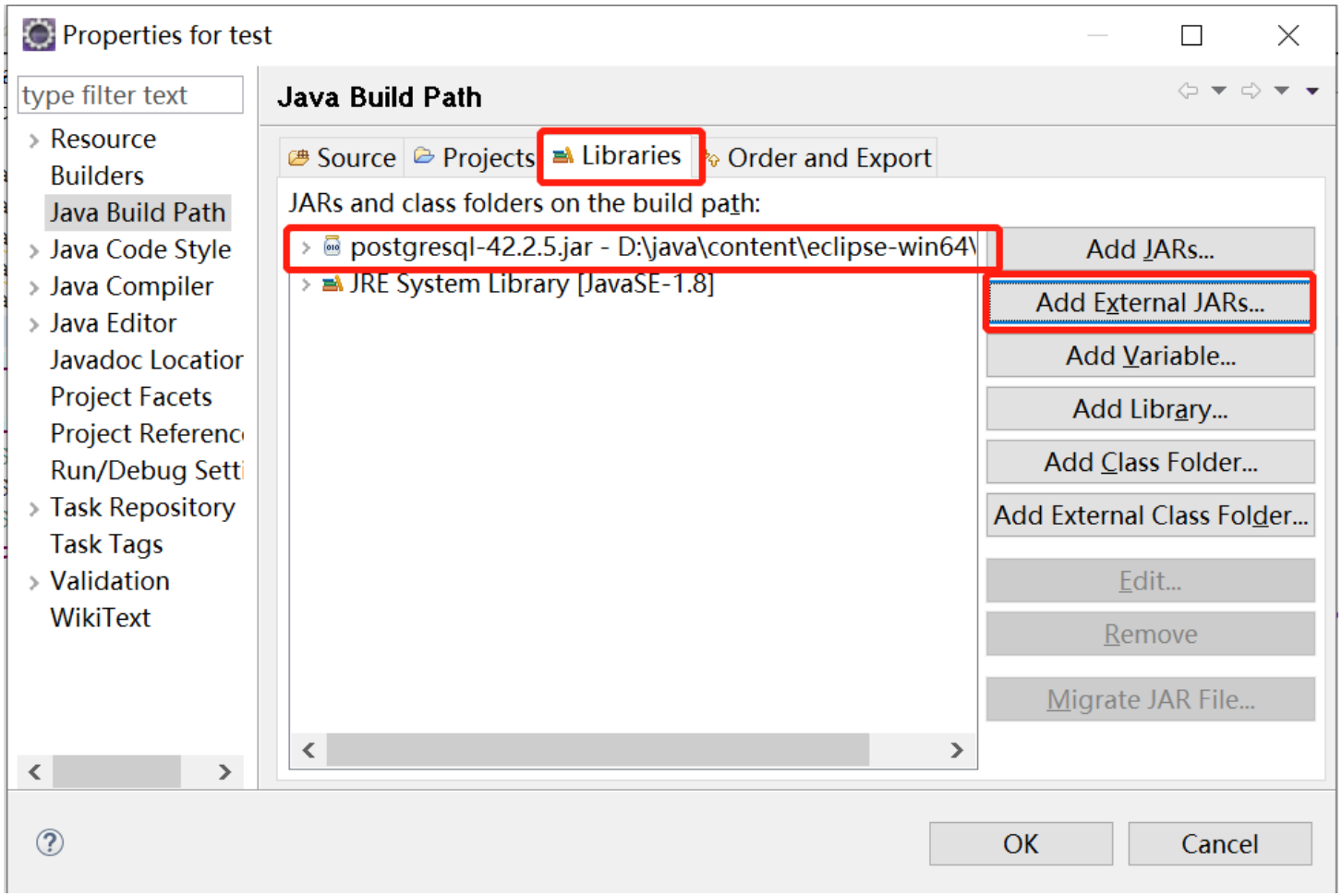
  - 安装完毕！

Java应用访问数据库—新建工程

- 新建java project——test
- 在test目录下新建lib文件夹
  - 将pg驱动文件 postgresql-42.2.5.jar复制过来

Java应用访问数据库—新建工程

- 右击test - Build Path – libraries – add external JARs 添加postgresql-42.2.5.jar



Java应用访问数据库—编写程序

- 新建class文件
  - 在这个文件中编写程序，连接数据库，增删改查
- JDBC应用程序

（1）加载JDBC驱动程序

（2）建立与数据库的连接

（3）进行数据库的访问

（4）关闭相关连接

Java应用访问数据库—编写程序

connect_pg.java

```
package test;    // java项目名字

public class connect_pg {    // class名

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }

}
```

Java应用访问数据库—连接数据库

```
try {
    Class.forName("org.postgresql.Driver");
    Connection con=DriverManager.getConnection
    ("jdbc:postgresql://localhost:5432/test","postgres","123
    456");
    //实际工作

    con.close();
} catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
```

加载驱动器类

要连接的数据库名字

关闭数据库连接

Java应用访问数据库—创建表

```
String sql_create="create table eee(eeid int primary key,
eeage int,dep_id int)";
```

关键1：sql语句

```
try {
        Class.forName("org.postgresql.Driver");
        Connection con  = DriverManage .getConnection
        ("jdbc:postgresql://localhost:5432/test",  "postgres",  "123456“);

        Statement stmt=con.createStatement();
        stmt.executeUpdate(sql_create);
        System.out.println("数据表创建成功");

        stmt.close();
        con.close();
} catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
}
```

关键2：sql语句的执行

通过connection对象创建SQL语句执行对象

Java应用访问数据库—插入数据

```
String sql_insert="insert into eee select
num,floor(random()*10+15),floor(random()*30+1000) from
generate_series(1,100) as t(num)";
```

```
try {
        Class.forName("org.postgresql.Driver");
        Connection con  = DriverManage .getConnection
        ("jdbc:postgresql://localhost:5432/test",  "postgres",  "123456“);

        Statement stmt=con.createStatement();
        stmt.executeUpdate(sql_insert);
        System.out.println("插入数据完成");

        stmt.close();
        con.close();
} catch (Exception e) {
      // TODO Auto-generated catch block
       e.printStackTrace();
}
```

随机生成100条测试数据，
eeid为1~100，不重复，
eeage在15~24之间，可重复，
dep_id在1000~1029之间，可重复

Java应用访问数据库—预备语句

```
String sql_insert2="insert into eee values(?,?,?)";

PreparedStatement pstmt=con.prepareStatement(sql_insert2);
pstmt.setInt(1, 101);
pstmt.setInt(2, 20);
pstmt.setInt(3, 1001);

pstmt.executeUpdate();
pstmt.close();
```

设置参数

Java应用访问数据库—查询

```
String sql_select="select * from eee where eeid<10";
 try {
        Class.forName("org.postgresql.Driver");
        Connection con  = DriverManage .getConnection
        ("jdbc:postgresql://localhost:5432/test",  "postgres",  "123456“);

        Statement stmt=con.createStatement();
        ResultSet re=stmt.executeQuery(sql_select);
       while(re.next())
        {
            int id=re.getInt("eeid");
            int age=re.getInt("eeage");
           int did=re.getInt("dep_id");
            System.out.println("eeid:"+id+"\teeage:"+age+"\tdep_id:"+did);
        }
        re.close();
        stmt.close();
        con.close();
 } catch (Exception e) {
       // TODO Auto-generated catch block
        e.printStackTrace();
 }
```

主要内容

JDBC编程

- Java环境配置
- Java应用访问数据库

PL/pgSQL函数

- 程序的基本结构
- 基本语法
- 函数的编写

与基础sql的区别

- 基础sql：前几节课的内容（create，select，……）
- PL/pgSQL除了基础sql语句，还包含了**程序化**的元素，
  - 比如在PL/pgSQL中可以使用if/then/else语句和循环功能。
  - 可以轻松地执行SQL语句，甚至对SQL语句的结果进行循环操作

为什么在服务器中进行程序设计

- 从一个简单的例子开始
  - 需要实现：互换考生A和考生B的院系号
  - 简单的sql语句

- 要确保有id号为A的考生，还要确保有id号为B的考生，然后再分别更新A、B的院系号。
- 都用sql语句来确认，很麻烦！

为什么在服务器中进行程序设计

- 程序设计的方法就可以使得这些工作更加容易管理、更加安全和更加强健。
- 可以把计算、核对和数据操作全部放在一个用户定义的函数里面。
- 用户直接调用函数即可。

程序的基本结构

思考：
- create
- create or replace

```
create or replace function
example(id varchar, name varchar)
returns varchar
as
$$
********
begin
    ********
end;
$$
language plpgsql;
```

1 函数声明

2 函数返回值类型

变量声明，定义

编程

3 主体部分

4 编程语言

基本语法

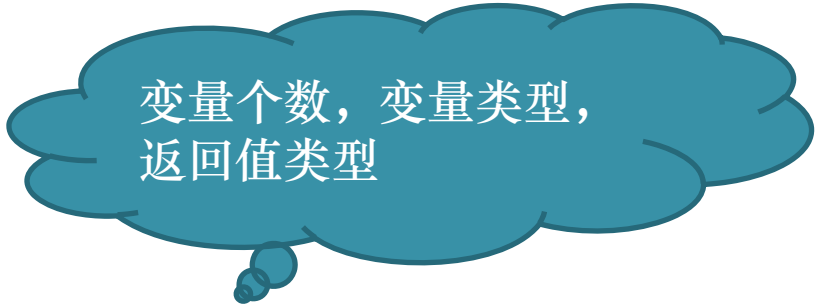
- 变量定义：
  - PL/pgSQL可以使用pgSQL中的所有数据类型来声明变量。
  - **Declare** age int;
- 变量赋值：
  - temp:=age;
- 如何把查询结果的值（单个结果）赋给变量
  - Select eeage from eee where eeid=2;
  - Select eeage **into age** from eee where erid=2;

注意：不是 int age！

以分号结尾

例子1 加法函数

- **要求：** 创建一个函数，计算整数a与b的和，并返回计算结果。



例子1 加法函数

```
create function add(a int,b int)
returns int
as
$$
declare sum int;
begin
    select a+b into sum;
    return sum;
end;
$$
language plpgsql;
```

	add integer
1	5

思考：还可以怎么写?

调用函数：

```
select add(2,3);
```

例子2 加法函数2

- **要求：** 创建一个函数，将eee表中所有考生年龄加2,，并返回考生人数。

例子2 加法函数2

- create function set_age(i_age int)
- returns int
- as
- $$
- begin
- update eee set eeage=eeage+i_age;
- return (select count(eeid) from eee);
- end;
- $$
- language plpgsql;

调用函数：

```
select set_age(2);
```

例子2 加法函数2

- create function set_age(i_age int)
- returns int
- as
- $$
- begin
- update eee set eeage=eeage+i_age;
- return (select count(eeid) from eee);
- end;
- $$
- language plpgsql;

调用函数：

```
select set_age(2);
```

思考：如果返回的不是单个结果，而是一个数据集呢？

例子2 加法函数2

删除函数：
drop function set_age(int);

- create function set_age(i_age int)
- returns int
- as
- $$
- begin
- update eee set eeage=eeage+i_age;
- return (select count(eeid) from eee);
- end;
- $$
- language plpgsql;

returns **setof** eee

return query select * from eee;

例子3 互换考生A和考生B的院系号

- 分析：
- 考生A是否存在……考生A不存在
- 考生B是否存在……..考生B不存在
- ……..操作成功

- 输入： ida int, idb int
- 函数： swap(ida int, idb int)
- 返回类型： returns text

```
create function swap(ida int, idb int)
returns text
as
$$
declare dep_a int;
declare dep_b int;
begin
    select dep_id into dep_a from eee where eeid=ida;
    if not found then
        return '考生A不存在';
    end if;
    select dep_id into dep_b from eee where eeid=idb;
    if not found then
        return '考生B不存在';
    end if;
    update eee set dep_id=dep_b where eeid=ida;
    update eee set dep_id=dep_a where eeid=idb;
    return '操作成功';
end;
$$
language plpgsql;
```

判断考生A、B是否存在

更新考生A、B的院系号

综合例子

- 创建一个PL/pgSQL函数，返回考生表中平均年龄最大的院系号和相应的平均年龄，并通过java程序调用。

综合例子

- create or replace function sele()
- returns table(did int,age numeric)
- as
- $$
- begin
- return query (with tt(did,avg_age) as (select dep_id,avg(eeage) from eee group by dep_id) select * from tt where avg_age=(select max(avg_age) from tt));
- end;
- $$
- language plpgsql;

综合例子

- 思考：如果不使用嵌套呢？