

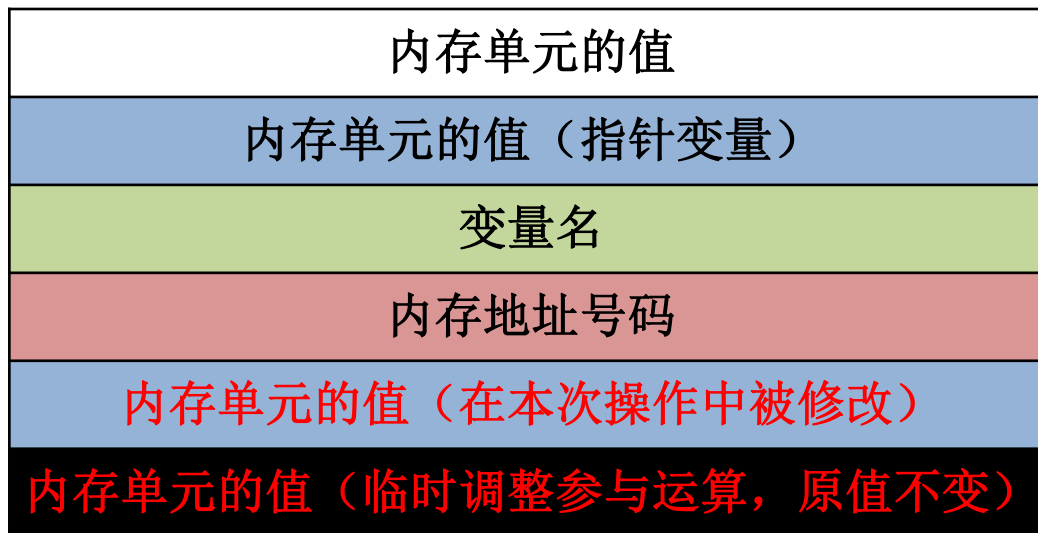
6-b1

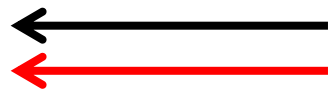
源程序代码

```
#include <iostream>
using namespace std;

int main()
{
    char *c[]={"John learn C++ language","Be well!","You","Not very"};
    char **p[]={c+3, c+2, c+1, c};
    char ***pp=p;
    cout << (**++pp);
    cout << (*--*++pp+4);
    cout << (*pp[-2]+3);
    cout << (pp[-1][-1]+2);
    cout << endl;
    return 0;
}
```

图例



 指向（指针运算符*操作方向）

内存分析

J	o	h	n	\32	l	e	...	a	n	g	u	a	g	e	\0
1000	1001	1002	1003	1004	1005	1006	...	1017	1018	1019	1020	1021	1022	1023	1024

B	e	\32	w	e	l	l	!	!	\0
1025	1026	1027	1028	1029	1030	1031	1032	1033	1034

Y	o	u	\0
1035	1036	1037	1038

N	o	t	\32	v	e	r	y	\0
1039	1040	1041	1042	1043	1044	1045	1046	1047

这4个字符串常量位于内存的静态存储区，只能读取，不能写入。

<i>c</i> [0]	1000	2000
<i>c</i> [1]	1025	2004
<i>c</i> [2]	1035	2008
<i>c</i> [3]	1039	2012

指针变量*c*[*i*]（1级）

基类型：char型 基类型占空间：1字节 值：-----
（实际上不存在，只是一种理解方式）

内存分析

$c[0]$	1000	2000
$c[1]$	1025	2004
$c[2]$	1035	2008
$c[3]$	1039	2012

指针变量 $c[i]$ (1级)
基类型: char型
基类型占空间: 1字节
值: -----
(实际上不存在, 只是一种理解方式)

c	2000	2100
---	------	------

指针变量c (2级)
基类型: 指向char型指针
基类型占空间: 4字节
值: 2000 (从此内存空间开始的4个字节被分配给用户使用)

$p[0]$	2012	2200
$p[1]$	2008	2204
$p[2]$	2004	2208
$p[3]$	2000	2212

指针变量 $p[i]$ (2级)
基类型: 指向char型指针
基类型占空间: 4字节
值: -----
(实际上不存在, 只是一种理解方式)

p	2200	2300
---	------	------

指针变量p (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节
值: 2200 (从此内存空间开始的4个字节被分配给用户使用)

pp	2200	2400
----	------	------

指针变量pp (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节
值: 2200

cout << (**++pp);

c[0]	1000	2000
c[1]	1025	2004
c[2]	1035	2008
c[3]	1039	2012

指针变量c[i] (1级)
基类型: char型
基类型占空间: 1字节
值: -----
(实际上不存在, 只是一种理解方式)

c	2000	2100
---	------	------

指针变量c (2级)
基类型: 指向char型指针
基类型占空间: 4字节
值: 2000 (从此内存空间开始的4个字节被分配给用户使用)

p[0]	2012	2200
p[1]	2008	2204
p[2]	2004	2208
p[3]	2000	2212

指针变量p[i] (2级)
基类型: 指向char型指针
基类型占空间: 4字节
值: -----
(实际上不存在, 只是一种理解方式)

p	2200	2300
---	------	------

指针变量p (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节
值: 2200 (从此内存空间开始的4个字节被分配给用户使用)

pp	2200	2400
----	------	------

指针变量pp (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节
值: 2200

等价: `cout << ((*(*pp=pp+1)));`

<i>c</i> [0]	1000	2000
<i>c</i> [1]	1025	2004
<i>c</i> [2]	1035	2008
<i>c</i> [3]	1039	2012

指针变量*c*[*i*] (1级)
基类型: **char**型
基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

<i>c</i>	2000	2100
----------	------	------

指针变量*c* (2级)
基类型: 指向**char**型指针
基类型占空间: 4字节

<i>p</i> [0]	2012	2200
<i>p</i> [1]	2008	2204
<i>p</i> [2]	2004	2208
<i>p</i> [3]	2000	2212

指针变量*p*[*i*] (2级)
基类型: 指向**char**型指针
基类型占空间: 4字节
(实际上不存在, 只是一种理解方式)

<i>p</i>	2200	2300
----------	------	------

指针变量*p* (3级)
基类型: 指向‘指向**char**型指针’的指针
基类型占空间: 4字节

<i>pp</i>	2204	2400
-----------	------	------

指针变量*pp* (3级)
基类型: 指向‘指向**char**型指针’的指针
基类型占空间: 4字节

等价: `cout << (*(*(pp=pp+1)));`

J	o	h	n	\32	l	e	...	a	n	g	u	a	g	e	\0
1000	1001	1002	1003	1004	1005	1006	...	1017	1018	1019	1020	1021	1022	1023	1024

B	e	\32	w	e	l	l	!	!	\0
1025	1026	1027	1028	1029	1030	1031	1032	1033	1034

Y	o	u	\0
1035	1036	1037	1038

N	o	t	\32	v	e	r	y	\0
1039	1040	1041	1042	1043	1044	1045	1046	1047

这4个字符串常量位于内存的静态存储区，只能读取，不能写入。

输出: You

指针变量`c[i]` (1级)
基类型: `char`型 基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

<code>c[0]</code>	1000	2000
<code>c[1]</code>	1025	2004
<code>c[2]</code>	1035	2008
<code>c[3]</code>	1039	2012


```
cout << (*--*++pp+4);
```

$c[0]$	1000	2000
$c[1]$	1025	2004
$c[2]$	1035	2008
$c[3]$	1039	2012

指针变量 $c[i]$ （1级）
基类型：char型
基类型占空间：1字节
（实际上不存在，只是一种理解方式）

c	2000	2100
---	------	------

指针变量c（2级）
基类型：指向char型指针
基类型占空间：4字节

$p[0]$	2012	2200
$p[1]$	2008	2204
$p[2]$	2004	2208
$p[3]$	2000	2212

指针变量 $p[i]$ （2级）
基类型：指向char型指针
基类型占空间：4字节
（实际上不存在，只是一种理解方式）

p	2200	2300
---	------	------

指针变量p（3级）
基类型：指向‘指向char型指针’的指针
基类型占空间：4字节

pp	2204	2400
----	------	------

指针变量pp（3级）
基类型：指向‘指向char型指针’的指针
基类型占空间：4字节

1004

等价: `cout << ((*(*pp=pp+1)=*(*pp=pp+1)-1))+4);`

$c[0]$	1000	2000
$c[1]$	1025	2004
$c[2]$	1035	2008
$c[3]$	1039	2012

指针变量 $c[i]$ (1级)
基类型: `char`型
基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

c	2000	2100
-----	------	------

指针变量 c (2级)
基类型: 指向`char`型指针
基类型占空间: 4字节

$p[0]$	2012	2200
$p[1]$	2008	2204
$p[2]$	2000	2208
$p[3]$	2000	2212

指针变量 $p[i]$ (2级)
基类型: 指向`char`型指针
基类型占空间: 4字节
(实际上不存在, 只是一种理解方式)

p	2200	2300
-----	------	------

指针变量 p (3级)
基类型: 指向‘指向`char`型指针’的指针
基类型占空间: 4字节

pp	2208	2400
------	------	------

指针变量 pp (3级)
基类型: 指向‘指向`char`型指针’的指针
基类型占空间: 4字节

等价: `cout << ((*(*pp=pp+1)=* (pp=pp+1)-1))+4) ;`

J	o	h	n	\32	l	e	...	a	n	g	u	a	g	e	\0
1000	1001	1002	1003	1004	1005	1006	...	1017	1018	1019	1020	1021	1022	1023	1024

B	e	\32	w	e	l	l	!	!	\0
1025	1026	1027	1028	1029	1030	1031	1032	1033	1034

Y	o	u	\0
1035	1036	1037	1038

N	o	t	\32	v	e	r	y	\0
1039	1040	1041	1042	1043	1044	1045	1046	1047

这4个字符串常量位于内存的静态存储区，只能读取，不能写入。

输出: `\32learn\32C++\32language`
(为了表述清晰，把“空格”写成`\32`，下同)

指针变量`c[i]` (1级)
基类型: `char`型 基类型占空间: 1字节
(实际上不存在，只是一种理解方式)

<code>c[0]</code>	1000	2000
<code>c[1]</code>	1025	2004
<code>c[2]</code>	1035	2008
<code>c[3]</code>	1039	2012

cout << (*pp[-2]+3);

$c[0]$	1000	2000
$c[1]$	1025	2004
$c[2]$	1035	2008
$c[3]$	1039	2012

指针变量 $c[i]$ (1级)
基类型: char型
基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

c	2000	2100
---	------	------

指针变量c (2级)
基类型: 指向char型指针
基类型占空间: 4字节

$p[0]$	2012	2200
$p[1]$	2008	2204
$p[2]$	2000	2208
$p[3]$	2000	2212

指针变量 $p[i]$ (2级)
基类型: 指向char型指针
基类型占空间: 4字节
(实际上不存在, 只是一种理解方式)

p	2200	2300
---	------	------

指针变量p (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节

pp	2208	2400
----	------	------

指针变量pp (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节

等价: `cout << ((*(*pp-2))+3);`

<i>c</i> [0]	1000	2000
<i>c</i> [1]	1025	2004
<i>c</i> [2]	1035	2008
<i>c</i> [3]	1042	2012

指针变量*c*[*i*] (1级)
基类型: `char`型
基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

<i>c</i>	2000	2100
----------	------	------

指针变量*c* (2级)
基类型: 指向`char`型指针
基类型占空间: 4字节

<i>p</i> [0]	2012	2200
<i>p</i> [1]	2008	2204
<i>p</i> [2]	2000	2208
<i>p</i> [3]	2000	2212

指针变量*p*[*i*] (2级)
基类型: 指向`char`型指针
基类型占空间: 4字节
(实际上不存在, 只是一种理解方式)

<i>p</i>	2200	2300
----------	------	------

指针变量*p* (3级)
基类型: 指向‘指向`char`型指针’的指针
基类型占空间: 4字节

<i>pp</i>	2000	2400
-----------	------	------

指针变量*pp* (3级)
基类型: 指向‘指向`char`型指针’的指针
基类型占空间: 4字节

等价: `cout << ((*(*pp-2))+3);`

J	o	h	n	\32	l	e	...	a	n	g	u	a	g	e	\0
1000	1001	1002	1003	1004	1005	1006	...	1017	1018	1019	1020	1021	1022	1023	1024

B	e	\32	w	e	l	l	!	!	\0
1025	1026	1027	1028	1029	1030	1031	1032	1033	1034

Y	o	u	\0
1035	1036	1037	1038

N	o	t	\32	v	e	r	y	\0
1039	1040	1041	1042	1043	1044	1045	1046	1047

这4个字符串常量位于内存的静态存储区，只能读取，不能写入。

输出: very

<i>c</i> [0]	1000	2000
<i>c</i> [1]	1025	2004
<i>c</i> [2]	1035	2008
<i>c</i> [3]	1042	2012

指针变量*c*[*i*] (1级)
基类型: char型 基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

```
cout << (pp[-1][-1]+2);
```

$c[0]$	1000	2000
$c[1]$	1025	2004
$c[2]$	1035	2008
$c[3]$	1039	2012

指针变量 $c[i]$ (1级)
基类型: char型
基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

c	2000	2100
---	------	------

指针变量c (2级)
基类型: 指向char型指针
基类型占空间: 4字节

$p[0]$	2012	2200
$p[1]$	2008	2204
$p[2]$	2000	2208
$p[3]$	2000	2212

指针变量 $p[i]$ (2级)
基类型: 指向char型指针
基类型占空间: 4字节
(实际上不存在, 只是一种理解方式)

p	2200	2300
---	------	------

指针变量p (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节

pp	2208	2400
----	------	------

指针变量pp (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节

等价: `cout << ((*((*pp-1))-1))+2;`

<i>c</i> [0]	1000	2000
<i>c</i> [1]	1004	2004
<i>c</i> [2]	1035	2008
<i>c</i> [3]	1039	2012

指针变量*c*[*i*] (1级)
基类型: char型
基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

<i>c</i>	2000	2100
----------	------	------

指针变量*c* (2级)
基类型: 指向char型指针
基类型占空间: 4字节

<i>p</i> [0]	2000	2200
<i>p</i> [1]	2004	2204
<i>p</i> [2]	2000	2208
<i>p</i> [3]	2000	2212

指针变量*p*[*i*] (2级)
基类型: 指向char型指针
基类型占空间: 4字节
(实际上不存在, 只是一种理解方式)

<i>p</i>	2200	2300
----------	------	------

指针变量*p* (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节

<i>pp</i>	2200	2400
-----------	------	------

指针变量*pp* (3级)
基类型: 指向‘指向char型指针’的指针
基类型占空间: 4字节

1027

2204

等价: `cout << ((*((*pp-1))-1))+2);`

J	o	h	n	\32	l	e	...	a	n	g	u	a	g	e	\0
1000	1001	1002	1003	1004	1005	1006	...	1017	1018	1019	1020	1021	1022	1023	1024

B	e	\32	w	e	l	l	!	!	\0
1025	1026	1027	1028	1029	1030	1031	1032	1033	1034

Y	o	u	\0
1035	1036	1037	1038

N	o	t	\32	v	e	r	y	\0
1039	1040	1041	1042	1043	1044	1045	1046	1047

这4个字符串常量位于内存的静态存储区，只能读取，不能写入。

输出: \32well!!

<i>c</i> [0]	1000	2000
<i>c</i> [1]	1025	2004
<i>c</i> [2]	1035	2008
<i>c</i> [3]	1039	2012

指针变量*c*[*i*] (1级)
基类型: char型 基类型占空间: 1字节
(实际上不存在, 只是一种理解方式)

综上，输出为：

You learn C++ language very well!!