



清华大学

腾讯会议 ID: 217 380 912/ 2021
Tsinghua University

计算机程序设计基础

第11讲 指针

沈瑜 (010-62782951)
shenyu@tsinghua.edu.cn

清华大学电机系

2021.11.23





作业常见问题

1. 文件结尾判断不当

```
FILE*fp1,*fp2;
if((fp1=fopen("S1.txt","r"))==NULL)
{
    printf("无法打开此文件\n");
    exit(0);
}
if((fp2=fopen("S2.txt","w+"))==NULL)
{
    printf("无法打开此文件\n");
    exit(0);
}
int i=0, key=0,num=0,len=0;
char a[100000]={'\0'};
while(!feof(fp1))
{
    for(i=0;i<100000;i++)
    {
        a[i]=fgetc(fp1);
    }
}
```

1. 文件大小不定，浪费空间

```
while (!feof(in2))
{
    for (int i = 0; i < 10; i++)
    {
        while (1)
        {
            ch = fgetc(in2);
            if (ch == '[')
                break;
        }
        tiaoke[i][0] = ch;
        int i = 1;
        while (ch != '\n')
        {
            ch = fgetc(in2);
            tiaoke[i][j] = ch;
            j++;
        }
    }
}
```

1 行数不确定

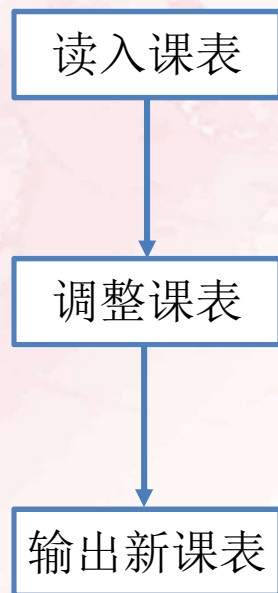
2 存在死循环的可能性



作业常见问题

2. 算法逻辑问题

- 一边读入课表数据一边修改数据
- [3]=[1] 导致原来周一的课程数据丢失



kebiao_old [][]

保留数据备份是个好习惯!

kebiao_new [][]

多测试几种情况，寻找隐藏的bug



作业常见问题

3.处理Windows文本文件的换行(\r\n)

- `fgets(buf,n,fp)` 如果n足够大,遇到'\r\n', 读入'\n',结尾补充'\0'
- `fgetc(fp)` 遇到'\r\n', 读入'\n', 同时文件指针后移2个字节
- `fscanf(fp,"%c",&ch)` 遇到'\r\n', 读入'\n', 同时文件指针后移2个字节
- `fscanf(fp,"%s",buf)`遇到'\r\n'时不读入, 文件指针停在'\r'前, '\r\n'被放在缓冲区, 需要根据下一fscanf语句具体分析。

如何验证: `ftell()`函数, debug查看内存/`printf`打印。

`fgets`和`fscanf("%s")`的区别: 字符串中有空格会导致只有部分字符被读入。





作业常见问题

fgets()与fgetc()

```
FILE* fp;
fp = fopen("KeBiao_input.txt", "r");
if (!fp) {
    return -1;
}

char info[100];
int d1, d2, n;
char cname[20], ch;
int t1, t2, t3;
fgets(info, 100, fp);
t1 = ftell(fp);

while (!feof(fp)) {
    t1 = ftell(fp);
    ch = fgetc(fp);
    t2 = ftell(fp);
    printf("ch=%c(%d) , \tt1=%d, \tt2=%d\n", ch, ch, t1, t2);
}
```

| 监视 1 | |
|------|---------------------------------|
| 名称 | 值 |
| ch | -52 '?' |
| info | 0x00befb0c "7 6 //共7天, 每天6节课\n" |
| t1 | 26 |

```
ch=?-42) ,      t1=70,  t2=71
ch="(34) ,      t1=71,  t2=72
ch=
(10) ,      t1=72,  t2=74
ch=[(91) ,      t1=74,  t2=75
ch=3(51) ,      t1=75,  t2=76
ch=,(44) ,      t1=76,  t2=77
ch=1(49) ,      t1=77,  t2=78
ch=](93) ,      t1=78,  t2=79
ch==(61) ,      t1=79,  t2=80
ch="(34) ,      t1=80,  t2=81
ch=?-52) ,      t1=81,  t2=82
ch=?-27) ,      t1=82,  t2=83
ch=?-45) ,      t1=83,  t2=84
ch=?-3) ,      t1=84,  t2=85
ch="(34) ,      t1=85,  t2=86
ch=
(10) ,      t1=86,  t2=88
ch=[(91) ,      t1=88,  t2=89
ch=3(51) ,      t1=89,  t2=90
```



作业常见问题

```
printf("\n *****fscanf(%%c)***** \n");
```

```
rewind(fp);
```

```
while(!feof(fp))
```

```
{
```

```
    t1=ftell(fp);
```

```
    n=fscanf(fp, "%c",&ch);
```

```
    t2=ftell(fp);
```

```
    printf("ch=%c(%d) , \tt1=%d, \tt2=%d\n", ch, ch, t1, t2);
```

```
}
```

```
ch=?-17) ,      t1=143, t2=144
```

```
ch="(34) ,      t1=144, t2=145
```

```
ch=
```

```
(10) ,      t1=145, t2=147
```

```
ch=[(91) ,      t1=147, t2=148
```

```
ch=5(53) ,      t1=148, t2=149
```

```
printf("\n *****fscanf(%%s)***** \n");
```

```
rewind(fp);
```

```
while(!feof(fp))
```

```
{
```

```
    t1=ftell(fp);
```

```
    n=fscanf(fp, "%s",cname); // 读到'\r'终止, '\n'被放在缓冲区
```

```
    t2=ftell(fp);
```

```
    //fscanf会自动跳过开头的非空字符'\n', 以下语句可以省略
```

```
    fscanf(fp, "%c",&ch);
```

```
    t3=ftell(fp);
```

```
    printf("%s , \tt1=%d, \tt2=%d, \tt3=%d\n", cname, t1, t2, t3);
```

```
}
```

```
*****fscanf(%%s)*****
```

```
7 ,      t1=0,      t2=1,      t3=2
```

```
6 ,      t1=2,      t2=4,      t3=5
```

```
//共7天, 每天6节课,      t1=5,      t2=24,      t3=26
```

```
[1, 2]="体育",      t1=26,      t2=38,      t3=40
```

```
[1, 4]="线性代数",      t1=40,      t2=56,      t3=58
```

```
[2, 4]="微积分",      t1=58,      t2=72,      t3=74
```

```
5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 1935, 19
```



作业常见问题

4. 不会解析文件 [1, 2]="体育" [1]=[3]

[7]=[]

- 使用fscanf 函数 "[%d,%d]=%s"

```
printf("\n *****fscanf ([%d,%d]=%s) ***** \n");
rewind(fp);
fgets(info, 100, fp);
while(!feof(fp))
{
    fscanf(fp, "[%d,%d]=%s", &d1, &d2, cname, 20);
    printf("[%d,%d]=%s\n", d1, d2, cname);
}
```

```
while(!feof(fp))
{
    fscanf(fp, "[%d,%d]=%s", &d1, &d2, cname, 20);
    ch=fgetc(fp); //可以发现读到了'\n'
    printf("[%d,%d]=%s\n", d1, d2, cname);
}
```

```
*****fscanf([%d,%d]=%s)*****
[1, 2]="体育"
[1, 2]="体育"
[1, 2]="体育"
[1, 2]="体育"
[1, 2]="体育"
[1, 2]="体育"
[1, 2]="体育"
```

```
*****fscanf([%d,%d]=%s)*****
[1, 2]="体育"
[1, 4]="线性代数"
[2, 4]="微积分"
[3, 1]="体育"
[3, 3]="C语言"
[3, 4]="英语"
[4, 1]="微积分"
[5, 2]="英语"
[5, 4]="线性代数"
[7, 2]="C语言"
[7, 2]="C语言"
[7, 2]="C语言"
```




作业常见问题

5. 多余的换行

注意检测fscanf的返回值，是实际读取的数据个数
若返回值为EOF（-1），表明读错误；

```
7 6 //共7天，每天6节课
[1,2]="体育"
[1,4]="线性代数"
[2,4]="微积分"
[3,1]="体育"
[3,3]="C语言"
[3,4]="英语"
[4,1]="微积分"
[5,2]="英语"
[5,4]="线性代数"
[7,2]="C语言"
```

```
00000000h: 37 20 20 36 20 20 2F 2F B9 B2 37 CC EC A3 AC C3 ; 7 6 //共7天，?
00000010h: BF CC EC 36 BD DA BF CE 0D 0A 5B 31 2C 32 5D 3D ; 刻?节课..[1,2]=
00000020h: 22 CC E5 D3 FD 22 0D 0A 5B 31 2C 34 5D 3D 22 CF ; "体育"..[1,4]="?
00000030h: DF D0 D4 B4 FA CA FD 22 0D 0A 5B 32 2C 34 5D 3D ; 咿源 ?..[2,4]=
00000040h: 22 CE A2 BB FD B7 D6 22 0D 0A 5B 33 2C 31 5D 3D ; "微积分"..[3,1]=
00000050h: 22 CC E5 D3 FD 22 0D 0A 5B 33 2C 33 5D 3D 22 43 ; "体育"..[3,3]="C
00000060h: D3 EF D1 D4 22 0D 0A 5B 33 2C 34 5D 3D 22 D3 A2 ; 语言"..[3,4]="英
00000070h: D3 EF 22 0D 0A 5B 34 2C 31 5D 3D 22 CE A2 BB FD ; 语"..[4,1]="微积
00000080h: B7 D6 22 0D 0A 5B 35 2C 32 5D 3D 22 D3 A2 D3 EF ; 分"..[5,2]="英语
00000090h: 22 0D 0A 5B 35 2C 34 5D 3D 22 CF DF D0 D4 B4 FA ; "..[5,4]="线性代
000000a0h: CA FD 22 0D 0A 5B 37 2C 32 5D 3D 22 43 D3 EF D1 ; 数"..[7,2]="C语?
000000b0h: D4 22 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A ; ?.....
000000c0h: 0D 0A 0D 0A 0D 0A ; .....
```

注意fscanf()的返回值

```
[5,2]="英语" n=3
[5,4]="线性代数" n=3
[7,2]="C语言" n=3
[7,2]="C语言" n=0
[7,2]="C语言" n=0
[7,2]="C语言" n=0
```

```
while(!feof(fp))
{
    n=fscanf(fp, "[%d,%d]=%s", &d1, &d2, cname); //读到'\r'终止，'\n'被放在缓冲区
    ch=fgetc(fp); //可以发现读到了'\n'
    //注意观察fscanf的返回值
    //printf("[%d,%d]=%s\tn=%d\n", d1, d2, cname, n);
    if(n==3)
        printf("[%d,%d]=%s\n", d1, d2, cname);
    else
        break;
}
```

```
*****fscanf("[%d,%d]=%s)*****
[1,2]="体育"
[1,4]="线性代数"
[2,4]="微积分"
[3,1]="体育"
[3,3]="C语言"
[3,4]="英语"
[4,1]="微积分"
[5,2]="英语"
[5,4]="线性代数"
[7,2]="C语言"
请按任意键继续. . .
```




主要内容

- C程序设计举例
- 指针的概念
- 指针操作
- 指针的应用
 - 指针变量作为函数参数
 - 指针变量作为函数返回值

参考教材： 第8章、第9.3、9.4节





11.1 C程序设计举例

- 例1: 编写函数，交换两个字符的值

验证：输入参数为 'c', 'd', 输出为'd', 'c'



实现功能前先写验证程序

```
#include <stdio.h>
#include <assert.h>

#define DEBUG

int main()
{
#ifdef DEBUG
    char c='c', d='d';
    swap( c, d );
    assert( c=='d' );
#endif
    return 0;
}
```

编程好习惯

程序

```
#include <stdio.h>
#include <assert.h>

#define DEBUG

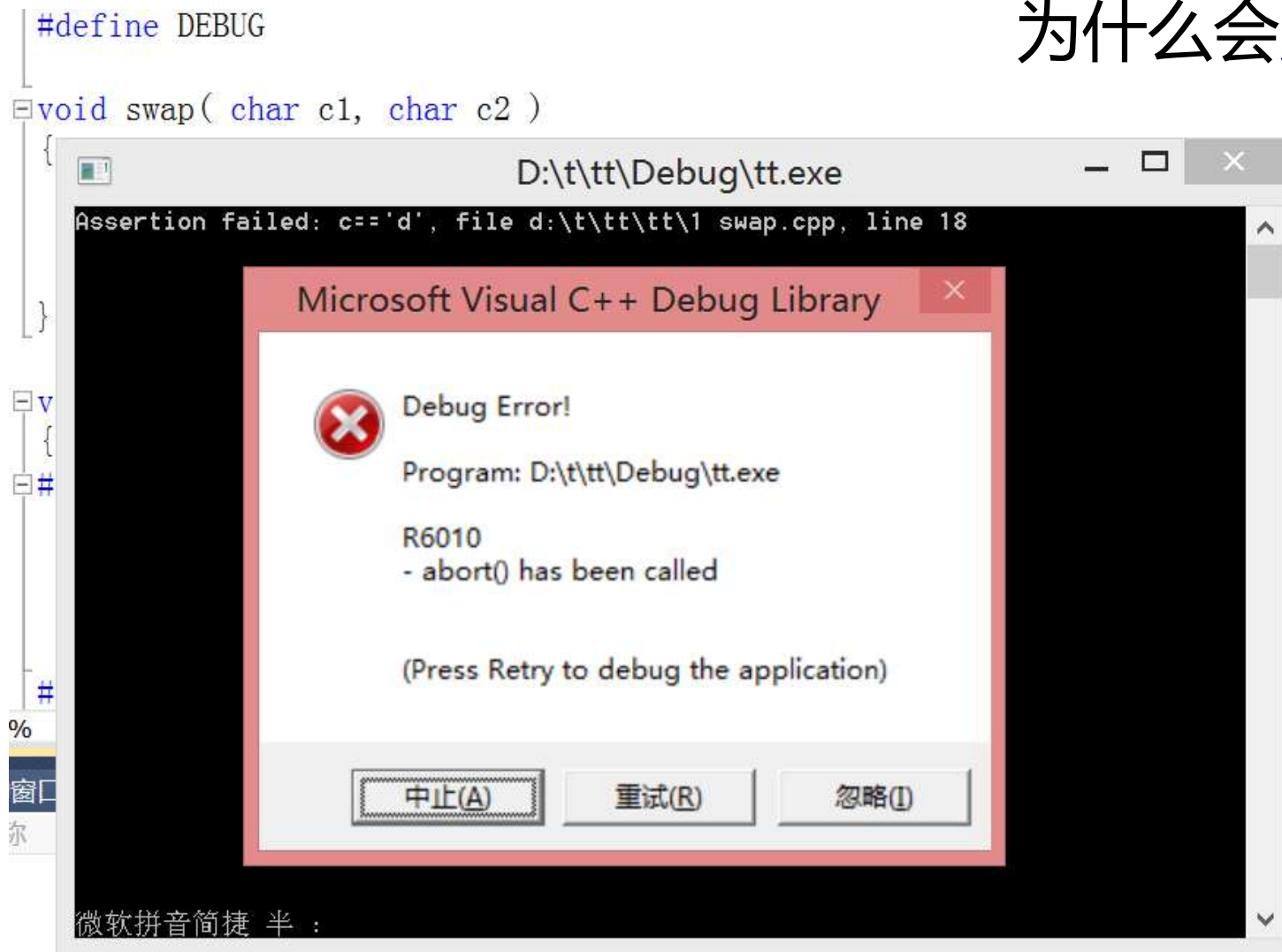
void swap( char c1, char c2 )
{
    char cTemp=c1;
    c1 = c2;
    c2 = cTemp;
}

int main()
{
#ifdef DEBUG
    char c='c', d='d';
    swap( c, d );
    assert( c=='d' );
#endif
    return 0;
}
```

这样写，有问题吗？

执行结果

为什么会**这样**?



原因分析

- 调用**swap**时，参数传递过程：
 - 将**c**赋给**c1**, **d**赋给**c2**
- 执行**swap**后：
 - **c1**与**c2**的值互相交换
- 问题：
 - 交换的是**c1**与**c2**的值，而不是**c**与**d**的值

```
#include <stdio.h>
#include <assert.h>

#define DEBUG

void swap( char c1, char c2 )
{
    char cTemp=c;
    c1 = c2;
    c2 = cTemp;
}

int main()
{
    #ifdef DEBUG
        char c='c', d='d';
        swap( c, d );
        assert( c=='d' );
    #endif
    return 0;
}
```

解决方案一：不使用子函数

- 不使用子函数，直接实现swap功能

```
#include <stdio.h>
#include <assert.h>

#define DEBUG

int main()
{
#ifdef DEBUG
    char c='c', d='d';
    char cTemp=c;
    c = d;
    d = cTemp;
    assert( c=='d' );
#endif
    return 0;
}
```



解决方案二：全局变量

- 将需要交换的变量定义为全局变量

```
#include <stdio.h>
#include <assert.h>

#define DEBUG

char c='c', d='d';

void swap( )
{
    char cTemp=c;
    c = d;
    d = cTemp;
}

int main()
{
    #ifdef DEBUG
        swap( );
        assert( c=='d' );
    #endif
    return 0;
}
```


解决方案三：？

- 方案一、方案二，可以吗？
- 可以。但是，swap函数是非常有用的功能。经常需要在程序中使用。
- 如果不将其封装成函数：
 - (1) 代码重复
 - (2) 程序可读性下降

那么，应该如何解决呢？



问题本质

- 希望:

在被调函数中修改主调函数中变量的值



这是可能的吗？

- 支持理由：
 - 主调函数中变量生存期长
 - 在被调函数执行过程中，主调函数中变量是存在的
- 不支持理由：
 - 主调函数与被调函数各有各的作用域
 - 在被调函数内看不到主调函数中变量



解决方案三：

- 间接使用主调函数中变量
 - 方法：将变量地址传过去，通过地址间接使用
 - 因为：主调函数中变量生存期长



如何传变量地址？

- 涉及内容
 - 变量地址表示方法？
 - 指针变量
 - 如何获得变量地址？
 - 取地址
 - 如何将获得的变量地址赋值给指针变量？
 - 指针赋值
 - 如何作为函数参数进行传递？
 - 指针作为参数
 - 如何利用获得的地址对变量进行操作？
 - 间接寻址



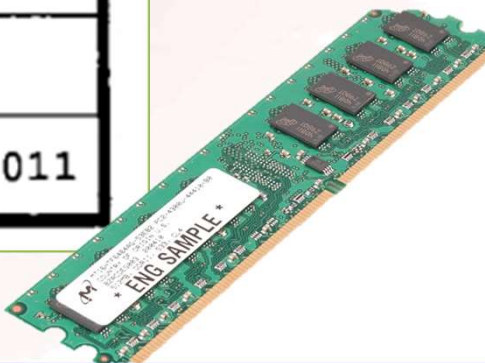


11.2 指针的概念

1. 回顾：内存

| 地址 | 内容 |
|-----|----------|
| 0 | 01010011 |
| 1 | 01110101 |
| 2 | 01110011 |
| 3 | 01100001 |
| 4 | 01101110 |
| | ⋮ |
| n-1 | 01000011 |

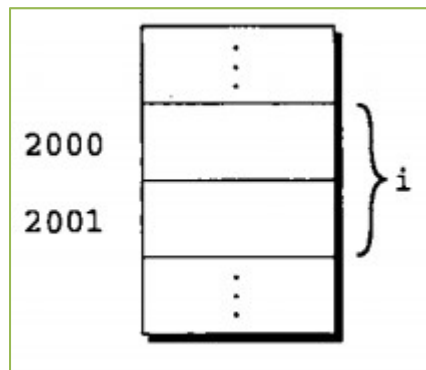
内存



- 程序中用到的变量和函数都转换成二进制位，存储在**内存芯片**中
- 内存芯片中空间的基本单位是字节（**8个bit**位）
 - **4G**芯片，相当与有**4G**个字节存储空间



回顾：指针就是地址



- 程序中每个变量占一个到多个字节，其第一个字节的位置称为**变量的地址**
- 虽然用数表示地址，但其取值范围与整数不同，所以我们用新的类型——**指针类型**存储地址
- 用指针变量**p**存储整型变量**i**的地址时，我们称：
 - ◆ **p指向i**
 - ◆ **p为整型指针**



回顾：空指针

| 地址 | 内容 |
|-----|----------|
| 0 | 01010011 |
| 1 | 01110101 |
| 2 | 01110011 |
| 3 | 01100001 |
| 4 | 01101110 |
| | ⋮ |
| n-1 | 01000011 |

- 值为**0**的指针称为空指针
 - 用**NULL**表示
 - 表示指针不指向内存中任何位置



2. 指针变量的定义

- 定义: 数据类型 *变量名

| 基本数据类型 | 对应指针类型 | 说明 |
|-------------------------|--------------------------|------|
| int data1, data2; | int *pData1, *pData2; | 整型指针 |
| char data1, data2; | char *pData1, *pData2; | 字符指针 |
| float data1, data2; | float *pData1, *pData2; | 实型指针 |
| double data1, data2; | double *pData1, *pData2; | 实型指针 |

data1, data2 为变量;
pData1, pData2为变量指针

指针变量定义有何特殊?

例1：使用辨析

```
void main()
{
    int* p1, p2;
}
```

相当于

```
void main()
{
    int *p1, p2;
}
```

或

```
void main()
{
    int (* p1), p2;
}
```

- p1 和 p2 都是指针吗？如果是，是什么类型的指针？

p1是整型指针

p2不是指针，是整数

*** 应与变量名写在一起**



例2：使用辨析2

```
void main()
{
    int *p1, data;
    double *p1, *9w;
    char *p$, ch, *c;
}
```

- 对吗？

- 错误1：*9w
 - 变量名不能以数字开头
- 错误2：*p\$
 - 变量名中只能包含数字、下划线与字母



例3：结构指针与文件指针

```
typedef struct
{
    double x, y;    //圆心
    double r;       //半径
} CIRCLE;

void main()
{
    CIRCLE c1={0, 0, 5};
    CIRCLE *p;
    FILE *f1;
}
```

- 对吗？
- p是指向什么类型的指针？
- f1呢？



11.3 指针操作

1. 取地址运算符

- 问题
 - `int a,b;`
 - 变量**a**, **b**在内存中存储地址是?
- 方法
 - 用**&**操作符
 - 在变量前加**&**符号, 求得变量地址
- 例
 - `int a,b;`
 - a的地址是: `&a`
 - b的地址是: `&b`



例1：取地址

```
void main() {  
    int    k = 9;  
  
    printf ("%d = %d\n", &k);  
    printf ("%x = %x\n", &k);  
    printf ("%X = %X\n", &k);  
    printf ("%o = %o\n", &k);  
    printf ("%p = %p\n", &k);  
}
```

```
%d = 5438440  
%x = 52fbe8  
%X = 52FBE8  
%o = 24575750  
%p = 0052FBE8
```

- &, 取变量地址
- %x, 以16进制输出整数(小写形式)
- %X, 以16进制输出整数(大写形式)
- %o, 8进制输出整数
- %p, 输出指针的值
- %, 输出一个%

例2：观察变量地址分布规律

- 有何规律？

```
#include <stdio.h>
void main()
{
    char    c1 = 'c', c2 = 'd', c3 = 'e';
    int     k = 9;
    float   f1 = 10.0, f2 = 0;
    double  d1 = 0.1, d2 = 0.2;

    printf("&c1 = %p\n", &c1);
    printf("&c2 = %p\n", &c2);
    printf("&c3 = %p\n", &c3);
    printf("&k  = %p\n", &k);
    printf("&f1 = %p\n", &f1);
    printf("&f2 = %p\n", &f2);
    printf("&d1 = %p\n", &d1);
    printf("&d2 = %p\n", &d2);
}
```

```
&c1 = 0102FEBF
&c2 = 0102FEB3
&c3 = 0102FEA7
&k  = 0102FE98
&f1 = 0102FE8C
&f2 = 0102FE80
&d1 = 0102FE70
&d2 = 0102FE60
```

地址分布规律

```
&c1 = 0102FEBF
&c2 = 0102FEB3
&c3 = 0102FEA7
&k  = 0102FE98
&f1 = 0102FE8C
&f2 = 0102FE80
&d1 = 0102FE70
&d2 = 0102FE60
```

```
&c1 = 010BFD2A
&c2 = 010BFD2B
&c3 = 010BFD29
&k  = 010BFD24
&f1 = 010BFD20
&f2 = 010BFD1C
&d1 = 010BFD14
&d2 = 010BFD0C
&c2 = 1
&c2 = -5
&c2 = 4
```

- 按变量定义顺序，由后往前连续分配空间；
- 空间最小分配单位：4个字节（8个字节）
 - 字符类型虽然只需要1个字节存储空间，但也被分配了12个
- Debug版本
- Release版本

例3：需要多大存储空间？

```
#include <stdio.h>
typedef struct
{
    char sex; // 'm', 'f'
    int age, height;
} PERSON;

void main()
{
    PERSON personA = {'m', 20, 180};
    PERSON personB = {'f', 16, 160};
    FILE file1;

    printf("&PersonA \t= %p\n",
           &personA);
    printf("&PersonB \t= %p\n",
           &personB);
    printf("PERSON size \t= %d\n",
           sizeof(PERSON) );
    printf("&file1 \t= %p\n", &file1);
    printf("FILE size \t= %d\n",
           sizeof(FILE) );
}
```

- **PERSON**需要多大的存储空间？
实际分配多大存储空间？
- 猜猜看，**FILE**结构需要多大存储空间？

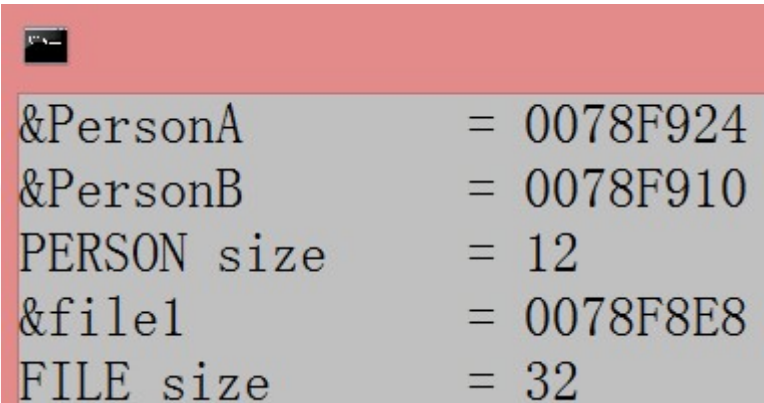


需要多大存储空间？

```
#include <stdio.h>
typedef struct
{
    char sex; // 'm', 'f'
    int age, height;
} PERSON;

void main()
{
    PERSON personA = {'m', 20, 180};
    PERSON personB = {'f', 16, 160};
    FILE file1;

    printf("&PersonA \t= %p\n",
           &personA);
    printf("&PersonB \t= %p\n",
           &personB);
    printf("PERSON size \t= %d\n",
           sizeof(PERSON) );
    printf("&file1 \t= %p\n", &file1);
    printf("FILE size \t= %d\n",
           sizeof(FILE) );
}
```



| | |
|-------------|------------|
| &PersonA | = 0078F924 |
| &PersonB | = 0078F910 |
| PERSON size | = 12 |
| &file1 | = 0078F8E8 |
| FILE size | = 32 |

- PERSON:
 - 需要：9个字节
 - 实际分配：12
- FILE： 32个字节

2.赋值与初始化

- 操作符 =
等号两侧操作数类型要一致

- 赋值

```
int *p1, k, *p2;
```

```
p1 = &k;
```

```
p2 = p1;
```

- 初始化

```
int k;
```

```
int *p1 = NULL;
```

```
int *p2 = &k;
```

```
void main()
{
    int    k = 9;
    char iAmH = 'h';

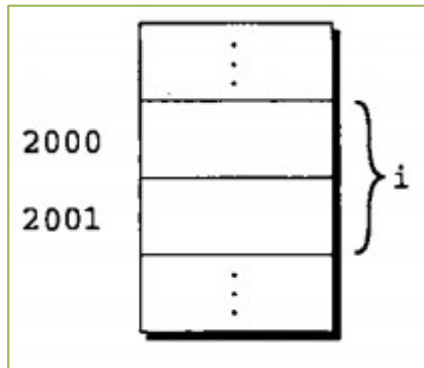
    int *p1 = &k;
    char *p2 = &iAmH;
    char *p3 = p1; //不对!

    printf("p1 = %p\n", p1);
    printf("p2 = %p\n", p2);
    printf("p3 = %p\n", p3);
}
```

- 指针类型不同，不能相互赋值



3. 间接寻址



变量 \leftrightarrow 一小块内存

- 寻址：
 - 找到与变量对应的内存位置，以便进行相应存取操作
- 直接寻址
 - 通过变量名，找到这一小块内存
- 例：将**10**存到**k**代表空间
 - `int k;`
 - `k = 10;`



间接寻址

- 间接寻址
 - 通过运用 *****运算符 操作指针变量，找到这一小块内存
- 例： 将**10**存到**q**指向空间
 - `int k; int *p = &k;`
 - `int *q = p;`
 - `*q = 10;`
- 问题： `int *q=p; *q=10;` 这两个语句中*号作用有何不同？



例：用 * 间接寻址

```
void main() {  
    int i;  
    int *p = &i;  
  
    // 1 存的操作  
    i = 10;  
    *p = 20;  
  
    // 2 取的操作  
    printf(" i = %d\n", i);  
    printf(" i = %d\n", *p);  
}
```

- 1、间接寻址
 - 通过**p**找到**i**，需要两步：先找到**p**，再根据**p**的内容来找到**i**
 - 所以是**间接寻址**
- 2、**i = 20**

***p：** 通过p中存储地址找到对应变量的地址

⇒ **p指向i**

例：指针必须正常初始化

```
#include <stdio.h>

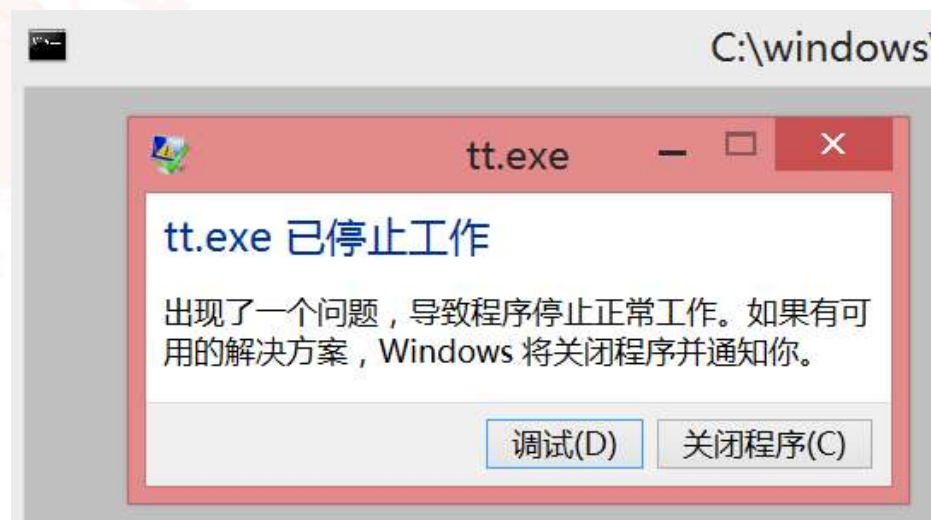
void main() {
    int i = 10, j = 20;
    int *p = NULL ;

    int k = i * j;
    *p = 30;
    int z = k * *p;

    printf(" k = %d\n", k);
    printf(" *p = %d\n", *p);
    printf(" z = %d\n", z);
}
```

• 区别：

- 间接寻址运算符：
单目运算符，后面
必须跟**指针变量**
- 乘法运算符：双目
运算符





11.4 应用

1. 应用一：指针变量作为函数参数

- 作用：在被调函数中修改主调函数中变量的值

例1：scanf

```
int n;  
printf("您要从1累加到几? ");  
scanf("%d", &n);
```

- scanf第2个参数传的是n的地址
- 为什么要这样做？
 - 需要在scanf中对n的值进行修改
 - 即：在被调函数中修改主调函数中变量的值



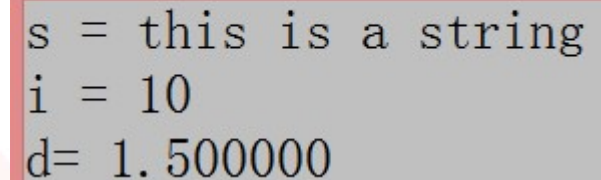
例：文件指针作为参数

```
#include <stdio.h>
#include <process.h>

void main( void )
{
    FILE *fp;
    int    i = 10;
    double d = 1.5;
    char   s[] = "this is a string";
    char   c = '\n';

    fp = fopen( "d:\\fprintf.out", "w" );
    fprintf( fp, "s = %s%c", s, c );
    fprintf( fp, "i = %d\n", i );
    fprintf( fp, "d= %f\n", d );
    fclose( fp );

    system( "type d:\\fprintf.out" );
}
```



```
s = this is a string
i = 10
d= 1.500000
```

- 作用
 - 在被调函数中修改主调函数中变量的值
 - 传递参数时，需要的内存空间更小

思考：如何让函数返回多个值？

- 一个函数只能有**0**个或**1**个返回值
 - 例： `int add(int a, int b);`
- 如果需要同时返回多个结果，该如何办？
 - 例：求一个实数的整数部分和小数部分
 - $x = k + y$
- `void decompose(double x, int *k, double *y);`



```
void decompose(double x, int *k, double *y) {  
    *k = (int)x;  
    *y = x - *k;  
}
```

```
void main() {  
    double x = 10.3, y;  
    int k;  
    decompose(x, &k, &y);  
    printf("x = %8.3f, k=%6d, y=%8.3f",x,k,y);  
}
```



2. 应用二：指针变量作为函数返回值

- 使用方法：与其他类型变量作为返回值一样

例1：文件指针作为返回值

```
FILE *fopen( const char *filename, const char  
*mode );
```



例：辨析

```
#include <stdio.h>
#include <assert.h>

#define DEBUG

int * accumulate(int nFrom, int nTo)
{
    int sum = 0;
    for(int i=nFrom; i<= nTo; i++)
        sum += i;
    return &sum;
}

void main( void )
{
    #ifdef DEBUG
        int *p = accumulate(1, 100);
        assert( *p == 5050 );
    #endif
}
```

- 能通过编译吗？

- 有**warning**

- 运行正确吗？

- **yes**

- 通过指针，主调函数间接使用了被调函数中局部变量

- 但是，被调函数中局部变量生存期短

- 当被调函数执行结束时，为局部变量分配空间即被操作系统收回

- 即指针指向的是已被收回的空间。可能会引发莫名其妙的错误！

结论：不应使用指针返回局部变量地址

小心！ 不要乱指哦



交换函数swap的实现

```
#include <stdio.h>
#include <assert.h>

#define DEBUG

void swap( char *p1, char* p2 )
{
    char *cTemp=p1;
    p1 = p2;
    p2 = cTemp;
}

int main()
{
#ifdef DEBUG
    char c='c', d='d';
    swap( &c, &d );
    assert( c=='d' && d=='c' );
#endif
    return 0;
}
```

对吗？

- 这样写对吗？
 - 不对
 - 交换的是**p1, p2**
 - 而不是**p1, p2**所指向的变量
 - 这样操作：对**c, d**还是没有影响



交换函数swap的实现

正解

```
#include <stdio.h>
#include <assert.h>

#define DEBUG

void swap( char *p1, char* p2 )
{
    char cTemp=*p1;
    *p1 = *p2;
    *p2 = cTemp;
}

int main()
{
    #ifdef DEBUG
        char c='c', d='d';
        swap( &c, &d );
        assert( c=='d' && d=='c' );
    #endif
    return 0;
}
```





小结

小结1：指针变量定义与基本操作

- 指针变量定义
- 指针基本操作
&, =, *





小结

小结2:

指针应用：指针作为参数与返回值

- 在被调函数中可以操作主调函数中变量
 - 有时必须这么做
 - 且：只需单向传递数据
 - 使用方便，效率高
 - 且：传递指针往往比传递数据需要的内存小
- 不应通过指针来返回被调函数中局部变量
 - 若一意孤行，





*指针的妙用

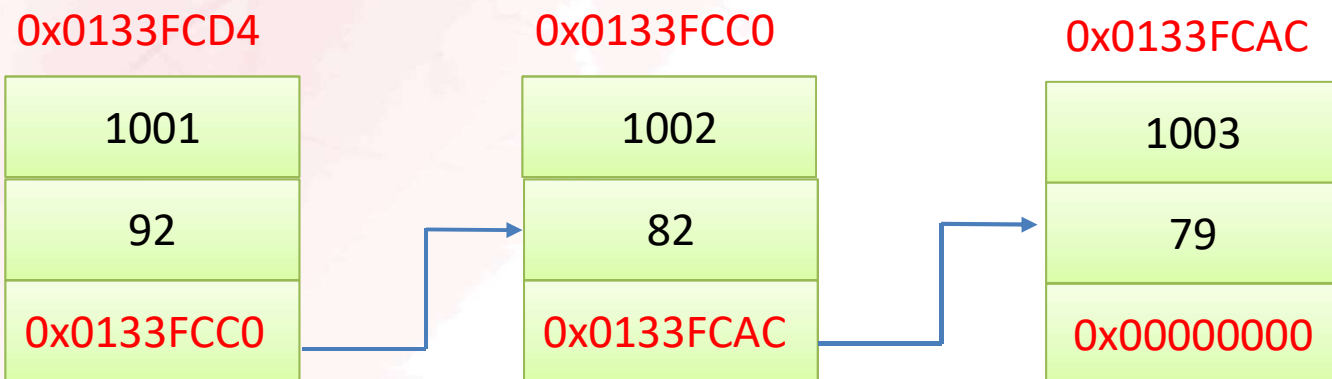
• 链表（教材9.4节）

```
struct student
{
    int num;
    int score;
    struct student* next;
};
```

```
int main()
{
    struct student stu1={1001, 92}, stu2={1002, 82}, stu3={1003, 79};
    struct student *pHead;

    pHead=&stu1;
    stu1.next=&stu2;
    stu2.next=&stu3;
    stu3.next=NULL;

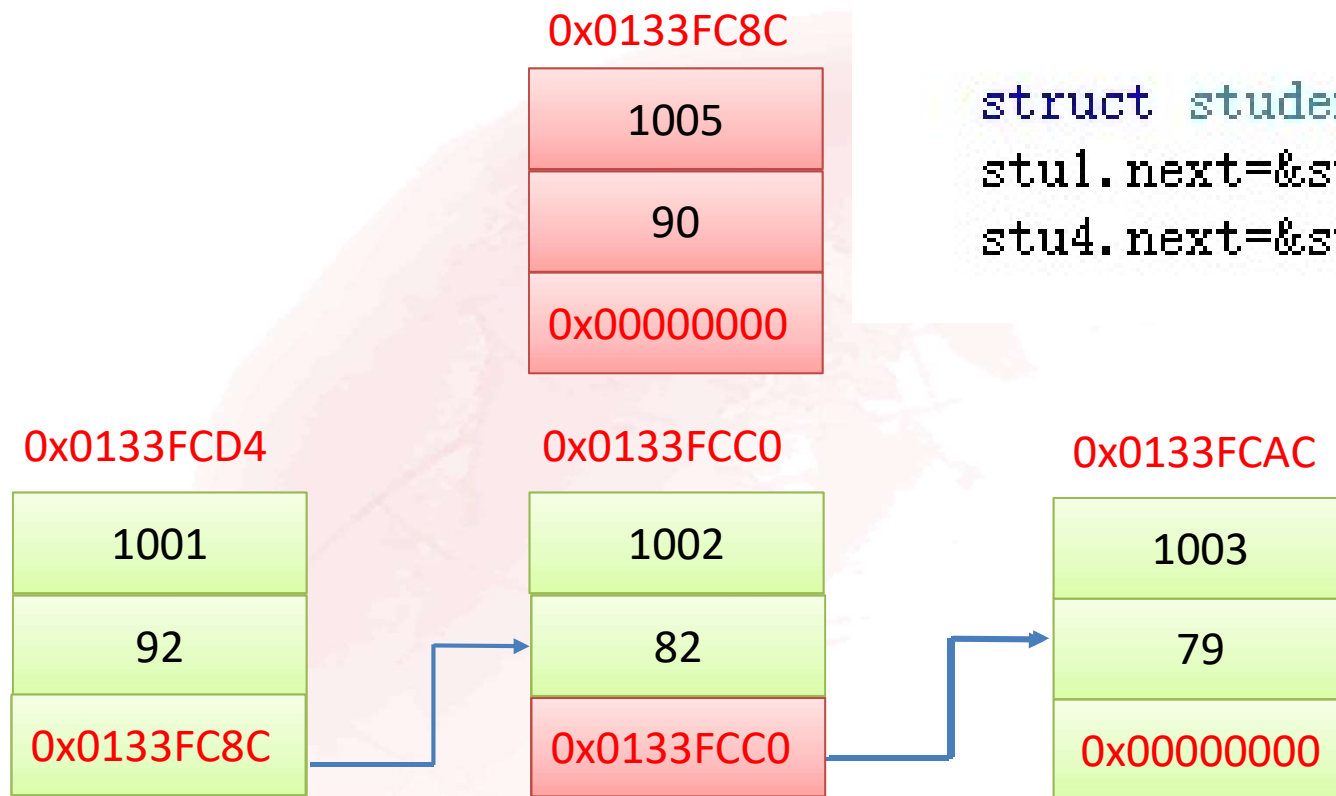
    return 0;
}
```





*指针的妙用

- 插入新节点



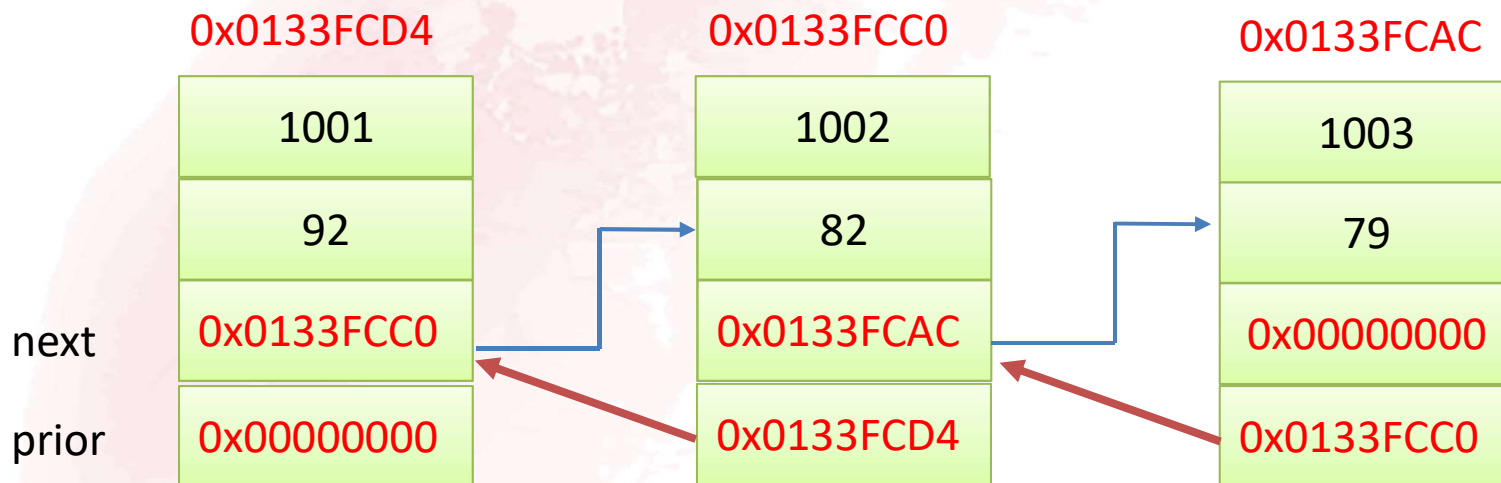
```
struct student stu4={1005, 90};  
stu1.next=&stu4;  
stu4.next=&stu2;
```



*指针的妙用

- 双向链表

```
struct student
{
    int num;
    int score;
    struct student* next;
    struct student* prior;
};
```



方便双向遍历





几点建议

- 读书读一本，反复看。不要看很多书，而是一本书看很多遍；
- 做题，反复做。不要做很多习题，而是一道题做很多遍。

贪多嚼不烂

此前各周的题目，请自己重新做！

