深 圳 大 学 实 验 报 告

课程名称：面向对象程序设计

实验项目名称：实验三 结构体与内存

学院：医学院

专业：生物医学工程

指导教师：李乔亮、邓云

报告人：陈焕鑫 学号：**2016222042** 班级：生工 2 班

实验时间：**2018.10.10**

实验报告提交时间：**2018.11.1**

教务部制

实验题目：

熟练掌握 C 语言中的结构体和联合体的使用，了解结构体和联合体中变量的内存分布

实验内容：

题目 1:分析以下程序，画出内存分布图，运行该代码，比较是否与分析结果一致；打开#pragma pack

(1)代码后再运行一遍，找出结果的变化并分析原因；将 struct 改为 union 后再做一次；

```
//#pragma pack(1)
struct node {
    char a;
    int b;
    short c;
};

void main() {
    struct node n;
    n.a = 0x01020304;
    n.b = 0x05060708;
    n.c = 0x010203040506;

    char * p = (char*)&n;
    for (int i = 0; i < sizeof(n); i++)
    {
        printf("p[%d] = 0x%x\n",i,p[i]);
    }
}
```

题目 2: 定义结构体表示学生的信息，至少包含名字，性别，年龄，学号 4 个信息,并完成以下功能

1) 在 main 函数中利用结构体数组定义 5 名学生，并初始化其中的信息（在程序内把学生信息存进去，不需要键盘输入） 。

2) 编写求平均年龄函数，返回所有学生的平均年龄, 请定义合适的函数形式。

3) 在 main 函数中调用求平均年龄函数，打印输出所有学生的信息，和平均年龄。

注：存储学生名字时，可采用 strcpy 函数。思考： 若学生人数不确定时，应该怎样分配空间来存储数据？

实验环境与程序代码及结构：

实验环境：win7 系统下的 Visual C++ 6.0

题目一：

原始程序如下所示：

```
#include <stdio.h>
//#pragma pack(1)

struct node {
  char a;
  int b;
  short c;
};

void main()
{
  struct node n;

  n.a = 0x01020304;
  n.b = 0x05060708;
  n.c = 0x010203040506;

  char * p = (char*)&n;

  for(int i = 0; i < sizeof(n); i++)
  {
    printf("p[%d] = 0x%x\n",i,p[i]);
  }
}
```

题目二：

程序代码如下所示：

```
#include <iostream>
#include <stdlib.h>

using namespace std;

#define random(x)  (rand() % x)

typedef struct
{
  char name[20];
```

```
  bool sex;
  short age;
  long num;
}Student;

float getAverage(Student *student, int num);

int main()
{
  Student stu[5];

  strcpy(stu[0].name, "Tim");
  strcpy(stu[1].name, "Amay");
  strcpy(stu[2].name, "John");
  strcpy(stu[3].name, "Lisa");
  strcpy(stu[4].name, "Jack");

  for(int i = 0; i < 5; i++)
  {
    stu[i].sex = (bool)(i % 2);
    stu[i].age = random(10) + 15;
    stu[i].num = random(10) + 2016222001;
  }

  for(int i = 0; i < 5; i++)
  {
    cout << "No." << i << " student's name is " << stu[i].name << "\t";

    if(stu[i].sex)
    {
      cout << "GIRL\t";
    }
    else
    {
      cout << "BOY\t";
    }

    cout << stu[i].age << " years old\t";
    cout << stu[i].num << endl;
  }

  float average = getAverage(stu, 5);

  cout << "The average of student's age is " << average;
```

```
  return 0;
}

float getAverage(Student *student, int num)
{
  int sum = 0;

  for(int i = 0; i < num; i++)
  {
    sum += student->age;
    student++;
  }

  return sum / num;
}
```

实验结果与分析：

题目一：

运行原程序之后出现的结果如图 1-1 所示：

```
p[0] = 0x4
p[1] = 0xffffffcc
p[2] = 0xffffffcc
p[3] = 0xffffffcc
p[4] = 0x8
p[5] = 0x7
p[6] = 0x6
p[7] = 0x5
p[8] = 0x6
p[9] = 0x5
p[10] = 0xffffffcc
p[11] = 0xffffffcc
Press any key to continue
```

图 1-1 未注释#pragma pack(1)运行结果

当我们把#pragma pack(1)注释掉之后，系统默认会将变量的地址对齐到变量字长的整数倍的地址处。这是因为编译器为了让程序跑的更快，减少 CPU 读取数据的指令周期，对结构体的存储进行了优化。实际上第一个 char 型成员虽然本来只有一个字节，但实际上却占用掉了 4 个字节，为的是让第二个 int 型成员的地址能够被 4 整除。同理，接下来的 short 类型的成员地址也能被 2 整除。为了能让结构体数组在更有效的被调用，整个结构体的大小也会被补齐到 4 的整数倍个字节。也就是，在 short 型变量的后面，编译器会自动补上 2 个空字节。因此，整个 node 占用了 12 个字节。如图 1-2

在打印输出的时候，char 类型的指针会逐个字节的读取内容，当读到空字节的时候，其内容就默认为 0xcc。

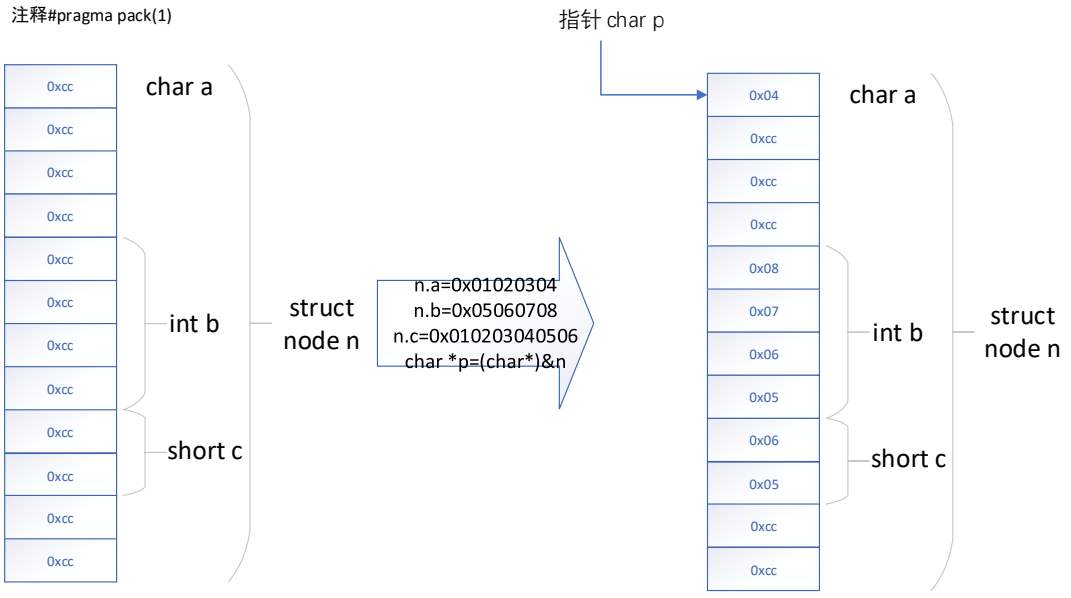

图 1-2 未使用#pragma pack(1)的内存图

打开#pragma pack(1)之后，结构体的边界对齐设置为 1 个字节，强制将结构体的数据连续排列。也就是所有数据在内存中是连续存储的。比较图 1-2 和图 1-3 中可以看出，原本 char 和 int 型变量之间用来补齐的空字节没有了，变量与变量之间是连续排列的。整个 node 只占用了 7 个字节。这样虽然能节约内存资源，但是在效率上会有所影响。

在输出的时候会将数据连续输出，不会读取到空字节。

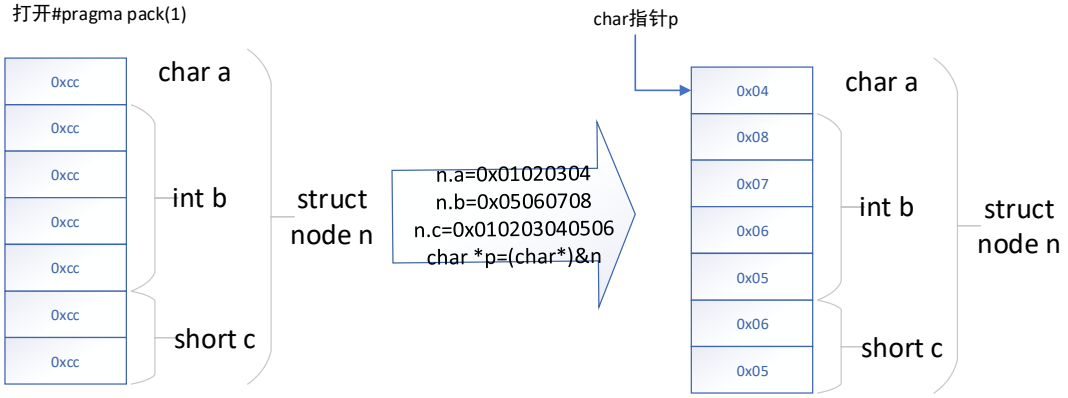

图 1-3 使用#pragma pack(1)的内存图

打开`#pragma pack(1)`，再一次运行程序，得到的结果如下所示

```
p[0] = 0x4
p[1] = 0x8
p[2] = 0x7
p[3] = 0x6
p[4] = 0x5
p[5] = 0x6
p[6] = 0x5
Press any key to continue
```

图 1-4 注释#pragma pack(1)运行结果

struct 和 union 的区别是：在存储多个成员信息时，编译器会自动给 struct 每个成员都分配存储空间，struct 可以存储多个成员信息，而 union 每个成员会共同使用同一个存储空间，只能存储最后修改的信息。它们虽然都是由多个不同的成员组成，但是在任何同一时刻，union 只存放一个被选中的成员，而结构体的所有成员都存在。从图 1-5 可以看出，整个 union 只占用 4 个字节的空间，而这四个字节的空间被 char、short 和 int 型的变量共同使用，且地址都相同。所以，在对 n 的成员进行一系列操作之后，往往只有最后一次操作是有效的（没有被操作得到的部分字节会被保留下来）。因此，可以看到，输出的结果是：p[0]=0x06, p[1]=0x05, p[2]=0x06, p[3]=0x05.

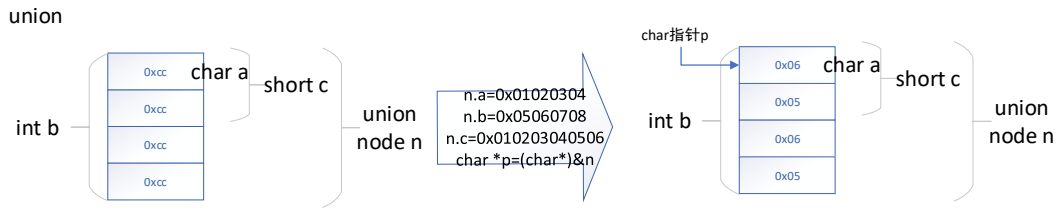

图 1-5 将 struct 改成 union 的内存图

将 struct 改为 union 之后运行程序得到的结果：

```
p[0] = 0x6
p[1] = 0x5
p[2] = 0x6
p[3] = 0x5
Press any key to continue
```

图 1-6 struct 改为 union 运行结果

题目二：

程序运行结果如图 1-7 所示

```
No.0 student's name is Tim      BOY     16 years old    2016222008
No.1 student's name is Amay     GIRL    19 years old    2016222001
No.2 student's name is John     BOY     24 years old    2016222005
No.3 student's name is Lisa     GIRL    23 years old    2016222009
No.4 student's name is Jack     BOY     17 years old    2016222005
The average of student's age is 19
请按任意键继续. . .
```

图 1-7

若人数不确定时，可以使用链表结构和动态内存分配，每增加一个学生，就使用 malloc 函数主动申请一块内存空间，存放新增加的学生的数据，然后链接到原来的存放学生数据的链表的末尾上，这样就可以灵活地调整学生的人数了。

指导教师批阅意见：

成绩评定：

指导教师签字：

年　　月　　日

备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。