

**实 验 报 告**

**课程设计名称：** 操作系统原理

**专 业 班 级 ：** 物联网工程1902班

**学 生 姓 名 ：** 王源

**学 号 ：** 201916070216

**指 导 教 师 ：** 廖庆荣

**课程实验时间：** 2021年4月19日

**实验二 主存的分配实现**

# 一、设计目的

1. 能实现操作系统主存的分配。

2. 能了解逻辑地址和物理块的对映转换。

# 二、设计要求与内容

1. 某虚拟存储器的用户空间共有x个页面，每页yKB，主存zKB。
2. 假定某时刻系统为用户的第0、1、2、3页分配的物理块号为5、10、4、7（这部分允许用户任意变化），而该用户作业的长度为w页，试输入逻辑地址转换成物理地址。例如输入0A5C，输出125C，并且可以判断缺页或越界。

# 设计原理：包含程序想法和做法说明

说明：实验环境：Windows操作系统、VS 2019/CMD\_g++、C/C++语言

由于是16进制，为了便于计算，我在程序计算中，转换为十进制来进行计算，题中没有给出每页的信息，让我们进行交互式地输入输出，通过地址转换规则便可进行物理地址的转换，其中考虑越界和缺页的情况，最终在输出前在转换为16进制即可。要想实现逻辑地址转换成物理地址，那么只需要将16进制的逻辑地址转换成二进制，并取高位（十进制页面数为2的几次方就取几位为页号）为页号，找到页号对应的块号，将10进制的块号转换成16进制的块号加上16进制的页内地址就可以求出物理地址了。

具体方法：

一、依据  
（1）计算十六进制使用公式  
逻辑地址=页号+页内地址  
物理地址=块号+页内地址  
  
【实现】  
逻辑地址转换为物理地址其实就是页号转换为块号。  
  
他们之间的页内地址是一样的。  
  
然后，我们要搞清楚页号、块号、页内地址怎么求的，然后通过公式带进去就可以求出逻辑/物理地址。  
  
页内地址的表达方式是2进制的，所以不管给的逻辑地址、物理地址都先转换为2进制。具体页内地址取几位，通过题目给的去求，如“每页为1KB”。  
  
（2）计算非十六进制使用公式  
物理地址=块号\*页内大小+页内地址  
页号=逻辑地址/页面大小字节=（取整数）  
页内地址=逻辑地址%页面大小字节=（取余数）

# 流程图

输入16进制逻辑地址

判断是否是大写字符

是

将16进制数转换成10进制数

再将转换好的10进制数转换成2进制

输出该二进制逻辑地址

取该二进制逻辑地址的前几位

将这几位二进制转换成10进制

找该页号对应的块号

取该二进制逻辑地址的前五位

no

该页号<=UserJobLen -1?

yes

cout(“页面大小超出页面长度”)；

找页号对应的块号

no

yes

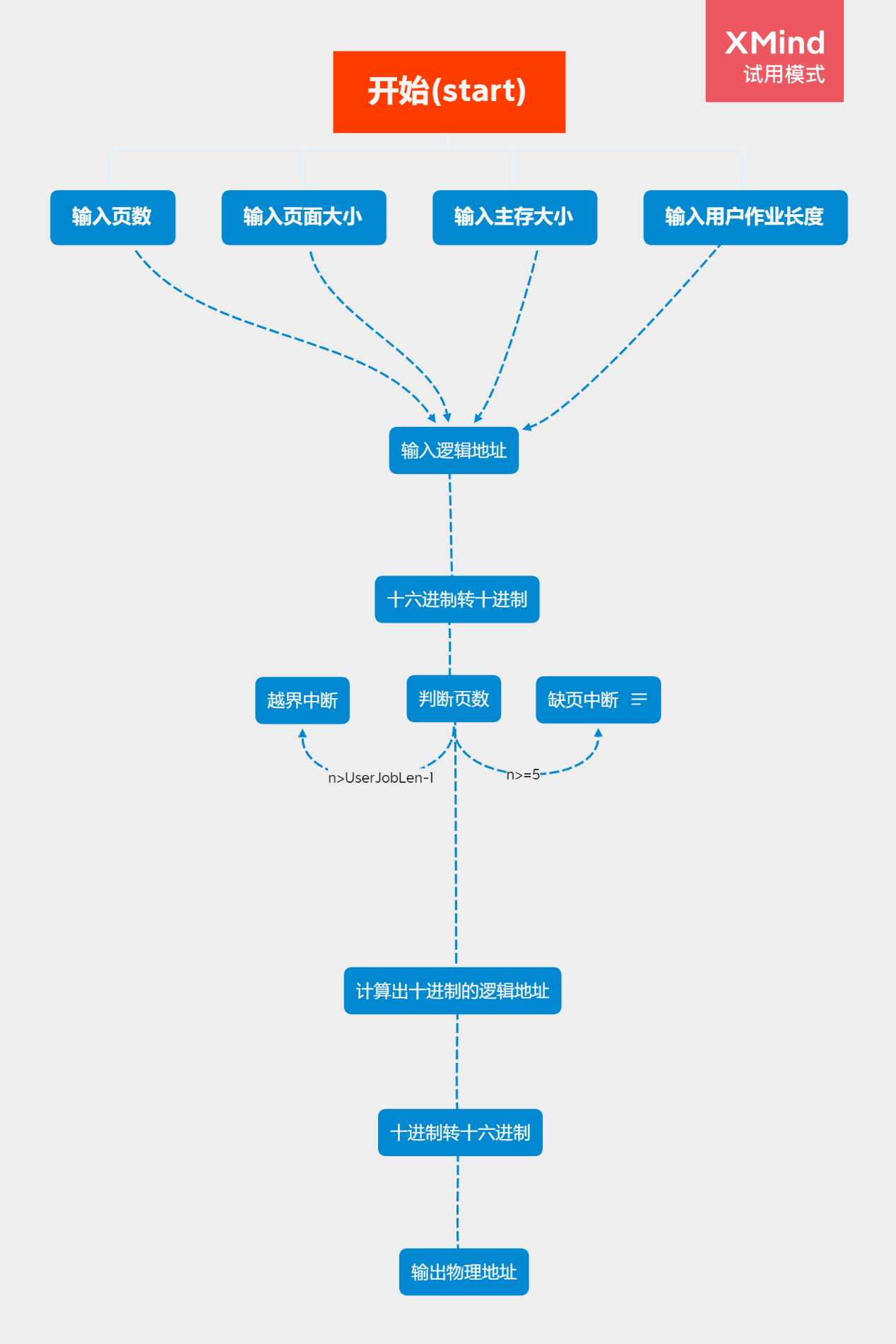
页号<=5?

将16进制的块号与后10位的页内地址组合

将10进制块号转换成16进制

cout(“page fault！”)；

输出16进制物理地址



# 源代码：程序一定要加上批注 /\* … \*/

源代码：

/\*

操作系统实验课OS\_02

物联网1902班

学号：201916070216

姓名：王源

\*/

allocate.h

#ifndef \_FUNC\_

#define \_FUNC\_

#include<iostream>

#include<string.h>

#include<stdlib.h>

#include<math.h>

#include<vector>

using namespace std; //标准命名空间展开

void info(); //学生信息

void func(int& PageCount, int& PageSize, int& MemorySize, int& count\_1, int& count\_2, int& count\_3, int& count\_4);

#endif

allocate.cpp

#include"allocate.h"

void info() //学生信息

{

cout << "\n “欢迎使用 主存的分配实现 小程序!”";

cout << "\n\n ---------------------------------------------\n";

cout << " | \n";

cout << " | 课程：操作系统实验课题 \n";

cout << " | \n";

cout << " | 题目：主存的分配实现 \n";

cout << " | \n";

cout << " | 作者：王源 \n";

cout << " | \n";

cout << " | 学号：201916070216 \n";

cout << " | \n";

cout << " ---------------------------------------------\n";

cout << "\tThe followings are my test examples:" << endl << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void func(int& PageCount, int& PageSize, int& MemorySize, int& count\_1, int& count\_2, int& count\_3, int& count\_4)

{ //交互式输出信息

cout << "请输入页面数量！：" << endl;

cin >> PageCount;

cout << "请输入页面大小(2的整数次方字节)！：" << endl;

cin >> PageSize;

cout << "请输入主存大小(2的整数次方字节)！：" << endl;

cin >> MemorySize;

while (PageCount > 1)

{

PageCount /= 2; //统计页号所占二进制位数

count\_1++;

}

while (PageSize > 1)

{

PageSize /= 2; //统计页内地址所占二进制位数

count\_2++;

}

count\_3 = count\_1 + count\_2;

// cout<<count\_3;

while (MemorySize > 1) //统计物理地址所占二进制位数

{

MemorySize /= 2;

count\_4++;

}

}

test.cpp

#include"allocate.h"

using namespace std; //标准命名空间展开

int main()

{

info(); //打印个人信息

cout << "we have already known the following page table\n" << endl; //显示页表

cout << "page number page frame" << endl;

cout << "0\t\t5\n1\t\t10\n2\t\t4\n3\t\t7\n4\t\t8\n5\t\t12\n" << endl; //输出页表

int PageCount = 0; //有多少个页面

int PageSize = 0; //每个页面的大小

int MemorySize = 0; //分配的主存的大小

int count\_1 = 0; //表示页号的二进制位数

int count\_2 = 0; //表示偏移量（页内地址） 的二进制位数

int count\_3 = 0; //表示用多少位二进制来表示逻辑地址

int count\_4 = 0;

int index = 0; //设置一个判断是否退出标志

do

{

func(PageCount, PageSize, MemorySize, count\_1, count\_2, count\_3, count\_4);

cout << "请输入用户作业长度！(必须>=5)：" << endl;

int UserJobLen = 0; //用户作业长度

cin >> UserJobLen;

//以下为十六进制转十进制

char Hex[20] = { 0 }; //存储十六进制字符串

//int b[count\_3] = { 0 }; //存储二进制字符串

vector<int> v; //开辟一个动态数组

v.resize(100); //提前设置好容量

int i = 0, len = 0, temp = 0;

int ret = 0;

out:

cout << "请输入16进制的逻辑地址：" << endl;

cin >> Hex;

for (size\_t i = 0; i < strlen(Hex); i++)

{

if (!((Hex[i] >= '0' && Hex[i] <= '9') || (Hex[i] >= 'A' && Hex[i] <= 'F') || (Hex[i] >= 'a' && Hex[i] <= 'f')))

{

cout << "你输入的16进制有问题，请检查后重新输入!" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

goto out;

}

}

cout << endl;

len = strlen(Hex); //统计字符串长度

for (i = 0; Hex[i] != '\0'; ++i)

{

switch (Hex[i]) //进行十六进制字符转换

{

case 'A': temp = 10; break;

case 'B': temp = 11; break;

case 'C': temp = 12; break;

case 'D': temp = 13; break;

case 'E': temp = 14; break;

case 'F': temp = 15; break;

default: temp = Hex[i] - '0'; break; //由于程序在计算机中执行，因此不必考虑十六进制以外的输入

}

ret += (temp \* pow(16, len - 1 - i));

}

int container\_1 = ret;

int j = 0;

while (container\_1 > 0) //十进制转二进制

{

//b[j] = container\_1 % 2;

v[j] = container\_1 % 2;

j++;

container\_1 /= 2;

}

cout << "逻辑地址二进制为：";

for (j = count\_3; j > 0; --j)

{

cout << v[j - 1]; //输出逻辑地址二进制

if ((j - 1) % 4 == 0)

{

cout << " ";

}

}

cout << endl;

//以下进行十进制分页计算

int page\_addr = 0, page\_num = 0;

int flag = 0;

page\_num = ret / 1024;

page\_addr = ret % 1024;

if (page\_num > UserJobLen - 1)

{ //判断是否存在特殊情况

cout << "越界中断！" << endl;

}

else if (page\_num > 5)

{

cout << "缺页中断！" << endl;

}

else

{

cout << "所在页数：" << page\_num << endl;

cout << "页内地址：" << page\_addr << endl;

flag = 1; //标记flag=1，以便下面进行计算

}

//以下进行十进制物理地址转换

if (flag == 1)

{

switch (page\_num) //逐一对应物理分区

{

case 0: page\_num = 5; break;

case 1: page\_num = 10; break;

case 2: page\_num = 4; break;

case 3: page\_num = 7; break;

case 4: page\_num = 8; break;

case 5: page\_num = 12; break;

default: break;

}

cout << "应分配的物理块号：" << page\_num << endl;

ret = page\_num \* 1024 + page\_addr;

int container\_2 = ret;

while (container\_2 > 0)

{

v[j] = container\_2 % 2;

j++;

container\_2 /= 2;

}

cout << "物理地址二进制为：";

for (j = count\_4; j > 0; --j) //输出物理地址

{

cout << v[j - 1];

if ((j - 1) % 4 == 0)

{

cout << " ";

}

}

cout << endl;

//以下为十进制转十六进制

i = 0;

int num[10];

char hex[16] = { '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F' };//标注十六进制字符

while (ret > 0) //计算十六进制中

{

num[i++] = ret % 16;

ret = ret / 16;

}

cout << "物理地址：";

for (i = i - 1; i >= 0; i--) //逐一对应输出十六进制字符

{

temp = num[i];

cout << hex[temp];

}

cout << endl;

}

out1:

cout << endl;

cout << "你是否要继续内存分配？如要继续，请输入enter，退出请按exit" << endl;

char str[10] = { 0 };

cin >> str;

if (strcmp(str, "enter") == 0)

{

index = 1;

}

else if (strcmp(str, "exit") == 0)

{

index = 0;

}

else

{

cout << "输入有误，请重新输入!:" << endl; //输入容错处理

goto out1;

}

} while (index); //判断是否退出程序

system("pause");

return 0;

}

# 实验结果与说明：包含程序运行结果截图，结果要先显示出你自己的班级、学号、姓名。

程序运行结果截图：

gcc/g++：

before modify: (没有分文件编写)



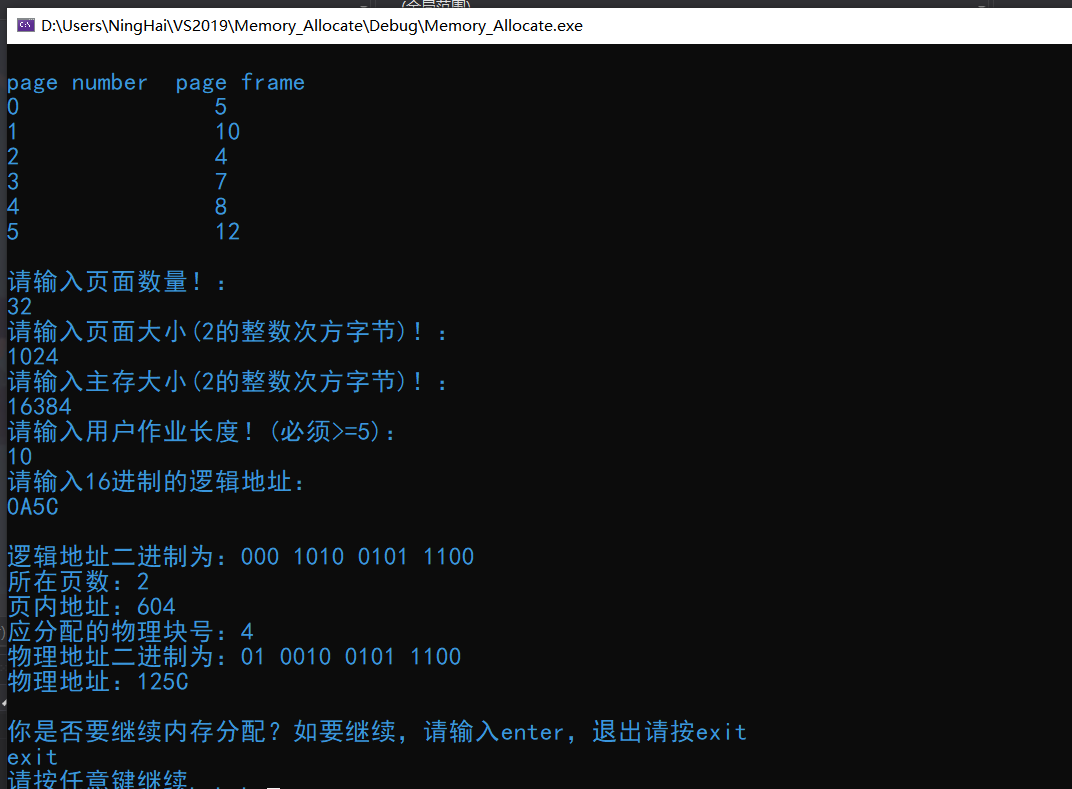


VS2019:

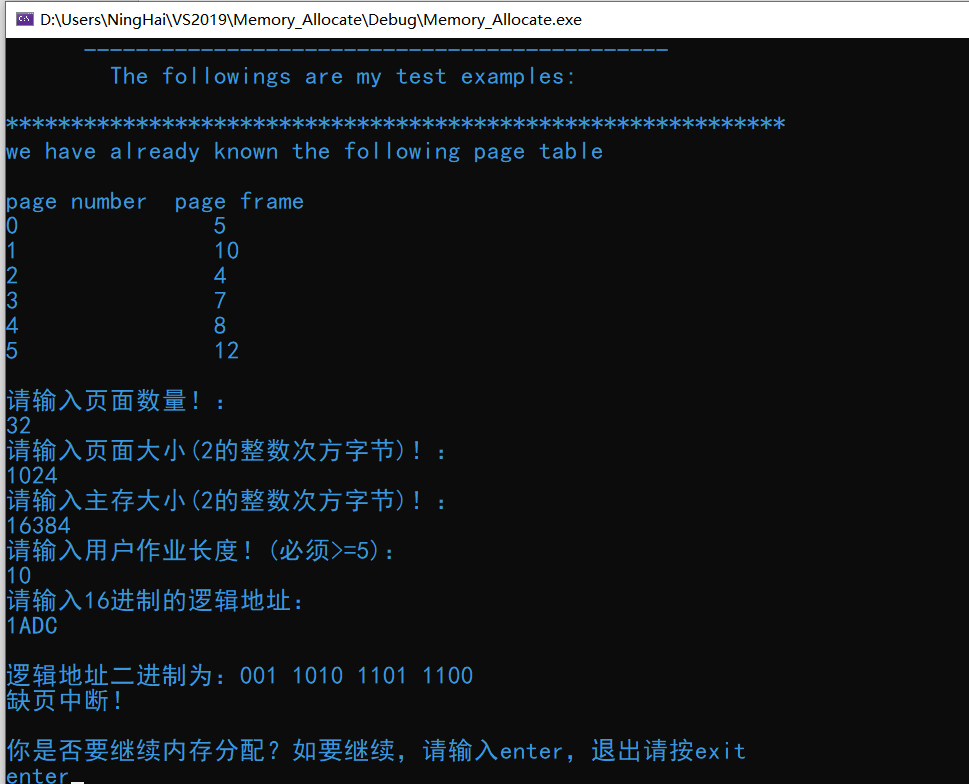
After modify: (使用了分文件编写)

1. 成功分配：





二．缺页中断：



三．越界中断：



# 我所遭遇的问题

1.我在页号和页内地址的存取以及块号的存放上出了一些问题，将逻辑地址转换成物理地址我的想法是将16进制的块号转换成二进制后，一个个让他们替换逻辑地址数组里对应的数，但是由于逻辑地址是15位，物理地址是14位，导致错位，我一直都是用替换数组的第n个数替换逻辑地址数组的第n个数，所以出现了错位的情况，我用笔重新演算了一遍代码后才明白是数组下标的问题，只用将替换数组的n＋1就可以实现了

2.在考虑进制转换时，因为涉及了十六进制，为了让代码长度尽可能短，想了很多方法，最终筛选出了我所呈现的方法。

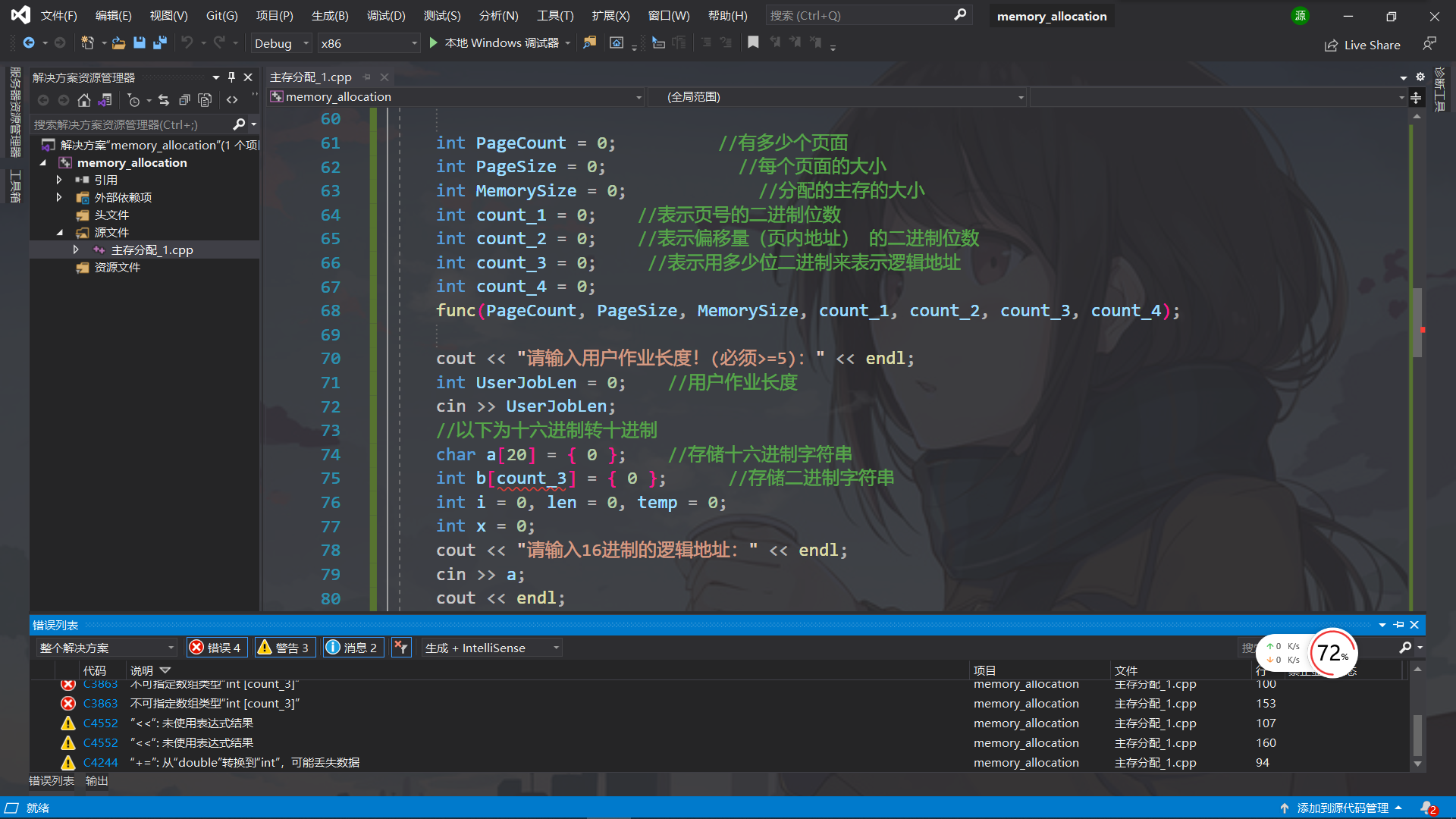
3.说实在的，起先看到实验题目是很懵的，因为自己对分页地址转换掌握的不太好，而本章的概念又特别多，很多名字相近的概念容易弄混，所以相比于代码实现，实际上概念上的理解上遇到的挫折更多。

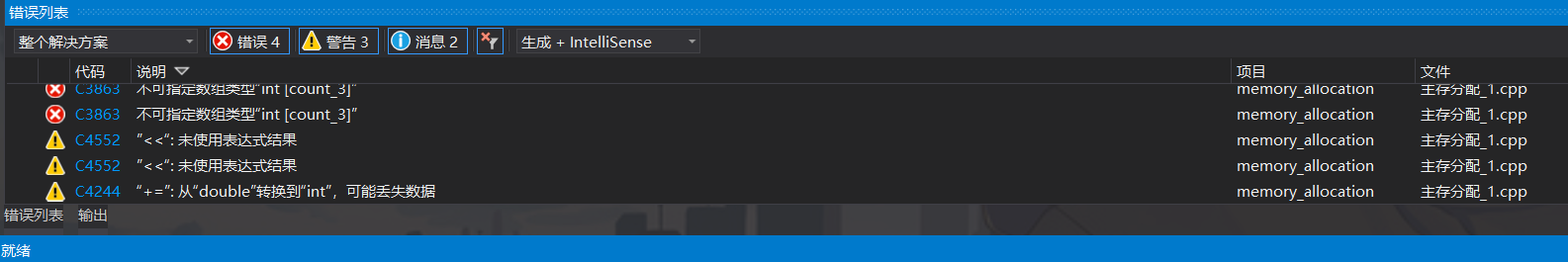
4.本实验主要在windows下VS2019环境实现，刚开始使用了变长数组，如下：

//以下为十六进制转十进制

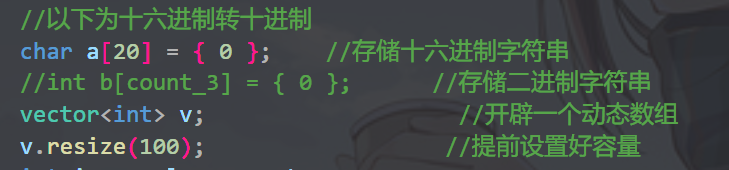
char a[20] = { 0 }; //存储十六进制字符串

int b[count\_3] = { 0 }; //存储二进制字符串

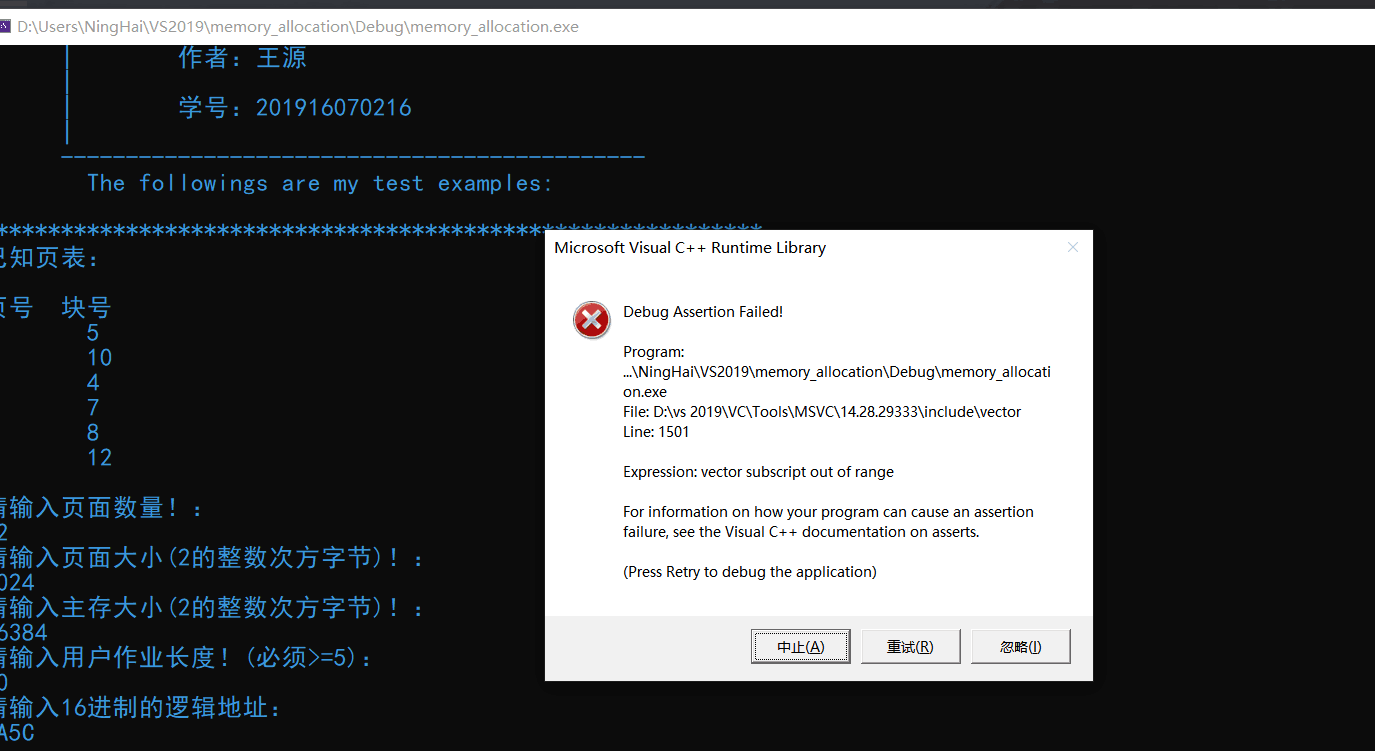
这里的count\_3是变量，变量在dev，g++，linux下是能够支持的，而我没想到VS2019不支持变长数组，所以在这里浪费了一些时间。



由于考虑到要让代码有更好的可移植性，我摒弃了那种做法，而转而用了STL里的vector容器类，vector支持动态变长数组，继而很好地解决了问题。



值得注意的是，使用vector之前要提前push\_back数据，或者提前开辟空间，如v.resize()，否则会报溢出错误（vector subscript out of range）。



# 收获、体会和建议：包含撰写程序的心得

通过此次实验，我对逻辑地址转换成物理地址的方法有了一个更深层次的理解，明白了逻辑地址转换成物理地址时候的注意事项，并且能够保证自己不会犯一些低级错误。这次实验的思路不难，但是十六进制转换成二进制的时候本想用平常8421对照法转换一下，但是实现起来就没那么简单，参考了一下资料，想起来十六进制先转十进制，再从十进制转换成二进制的方法比较容易实施。因此我觉得以前的基础知识我还是不扎实，我需要再多加回顾回顾。

总的来说：

1、充分掌握了新知识，了解了计算机内存的工作原理、分页规则以及逻辑地址和物理地址的转换机制。

2、通过实际编程进行地址转换，体会到了计算机工作的巧妙，以及先驱的智慧。

九．附件



# 主要参考文献

1. CSDN社区：《c++实现---十六进制转化为十进制 》

<https://blog.csdn.net/Fizz6018/article/details/107323868>

1. CSDN社区：《操作系统\_逻辑地址转换为物理地址》

<https://blog.csdn.net/sinat_34166518/article/details/83278553>

1. 汤小丹，梁红兵等 《计算机操作系统》(分页存储管理方式)

**截止日期：2021.04.21 Wed. 23:00**

**档案命名方式：OS02\_班级学号姓名**