**面向对象程序设计C++ 习题1**

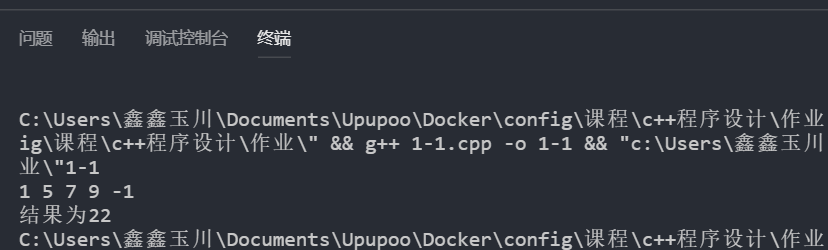
1. 课程的许多例子中使用到了流库，但流库的内容在后面第11章才会详细讲解，请大家提前学习相关内容，特别是与标准输入输出相关的流cin和cout的使用。
2. 练习不定次数的循环，以及输入流结束的判断（输入结束方式，请自行设定）。编写程序，从cin读取一组数，输出其和。

规定以-1为输入结束符

核心代码：

|  |
| --- |
| #include<iostream>  // cin读入一组数组,以-1结束  int main(){  int sum = 0;  int cur = 0;  do{  std::cin >> cur;  sum += cur;  } while (cur != -1);  std::cout << "结果为" << sum + 1;  return 0;  } |

运行结果图



**面向对象程序设计C++ 习题2**

1. 编程实现，首先编写4个子函数，分别进行两个数的加减乘除运算。在主函数中初始化一个长度为4的数组，数组内容分别包含指向这个4个函数的指针。从键盘输入3个数字，第1、2个数字表示运算参数，根据输入的第3个数字（为整数1-4），从数组中获取函数指针，并进行相应的运算，最终输出计算结果。

核心代码

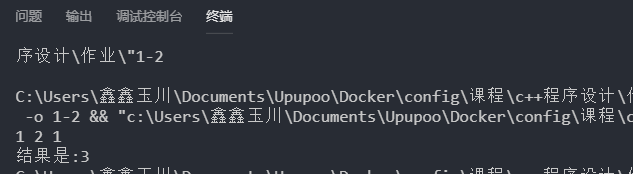
|  |
| --- |
| #include<iostream>  void plus(int a,int b){  std::cout<< "结果是:" << a + b;  return;  }  void subtract(int a,int b){  std::cout << "结果是:" << a - b;  return;  }  void multiply(int a,int b){  std::cout << "结果是:"<< a \* b;  return;  }  void divide(int a,int b){  std::cout << "结果是:" << a \* 1.0 / b;  return;  }  int main(){  int a, b;  int operateSymbol;  std::cin >> a >>b>> operateSymbol;  void (\*operates[4])(int a, int b) = {plus, subtract, multiply, divide};  if (1 <= operateSymbol && operateSymbol<=4){  operates[operateSymbol-1](a, b);  } else{  std::cout << "非法的运算输入";  }  return 0;  } |

1. 接第1题，添加子函数，根据输入参数（为整数1-4），返回指向4中运算的函数，修改主函数，实现与第2题相同的运行结果。

关键代码

|  |
| --- |
| typedef void (\*operate)(int,int);  operate fun(int operateSymbol)  {  operate res = nullptr;  switch (operateSymbol)  {  case 1:  res = plus;  break;  case 2:  res = subtract;  break;  case 3:  res = multiply;  break;  case 4:  res = divide;  break;  default:  break;  }  return res;  }  operate f = fun(operateSymbol);  f(a,b); |

结果图



1. 自学书上4.7节，补充完成第94页，4.7.5节中“2.链表的接口”中的未完成的接口函数，如push\_back等。

关键代码（带注释）

|  |
| --- |
| void push\_back(QUADPTR p)  {  Node \*tmp = new Node(); // 为加入的Node结构体分配一个新的内存  tmp->quad = p; // 将申请Node结构体的quad初始化为q  tmp->next = nullptr; // 将申请Node结构体的next初始化为NULL  if (tail == nullptr){ // 判断链表是否为空  head = tmp; //链表为空时，把head和tail都置为tmp  tail = tmp;  } else{  tail->next = tmp; // 将链表的尾部指针的next指向新申请的Node结构体  tail = tmp; // 将链表的尾部置为新申请的Node结构体  }  len += sizeof(Node);// 加上新加入的size  return;  } |

**面向对象程序设计C++ 习题3**

1. 学习书中5.8节，补充完成5.8.1节中矩形Rectangle、正方形Square、梯形Trapezoid和菱形Diamond等类的代码设计；完成5.8.2节中的List类中缺少的类成员函数。编写测试代码，随机push一些四边形到List中，然后遍历输出相关信息。

关键代码

|  |
| --- |
| quad.h  class Square  {  private:  Quadrangle quad;  int width,height;  public:  void create(std::string tag, int w, int h);  double area();  void draw(bool showResult = true);  Square(){  quad.tag = "Square";  }  }; |

|  |
| --- |
| list.h  void push\_back(QUADPTR p)  {  Node \*tmp = new Node(); // 为加入的Node结构体分配一个新的内存  tmp->quad = p; // 将申请Node结构体的quad初始化为q  tmp->next = nullptr; // 将申请Node结构体的next初始化为NULL  if (tail == nullptr){ // 判断链表是否为空  head = tmp; //链表为空时，把head和tail都置为tmp  tail = tmp;  } else{  tail->next = tmp; // 将链表的尾部指针的next指向新申请的Node结构体  tail = tmp; // 将链表的尾部置为新申请的Node结构体  }  len += sizeof(Node);// 加上新加入的size  return;  }  void itrator(){  if (tail == nullptr){  std::cout << "链表是空的"<<std::endl;  } else{  int count = 1;  Node\*tmp = head;  while(tmp !=tail){  std::cout << "链表第" << count << "个是" << tmp->quad->tag<<std::endl;  count++;  tmp = tmp->next;  }  std::cout << "链表第" << count << "个是" << tmp->quad->tag;  }  } |

|  |
| --- |
| main.cpp  #include "1-3-list.h"  int main() {  QUADPTR quad[5];  for (int i = 0; i < 5;i++)  {  quad[i] = new Quadrangle();  }  quad[0]->tag = "Parallelogram";  quad[1]->tag = "Rectangle";  quad[2]->tag = "Square";  quad[3]->tag = "Trapezoid";  quad[4]->tag = "Diamond";  List \*list = new List();  list->itrator();  for (int i = 0; i < 5;i++)  {  list->push\_back(quad[i]);  }  list->itrator();  return 0;  } |

运行结果图

