实验编号：实验二 实验名称：类的构建

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 班级 | 通信 2018-03 班 | | 学号 | 2018113752 | 姓名 | 谢博文 |
| 问题2.1 | 创建 class A 的三个对象，依次从输入给对象 a 和 b 的 x,y 赋值，并使得对象 c 中的  x=a.x+b.x , y=a.y+b.y，并从控制台输出。 | | | | | |
| 问题2.2 | #include "iostream"  using namespace std;  class A {  private:  int x, y;  public:  void setx(int num) { this->x = num; }; //函数重载 有num传入就使用num，没有就从cin获取  void sety(int num) { this->y = num; };  void setx() { cin >> this->x; };  void sety() { cin >> this->y; };  int getx() {  return this->x;  };  int gety() {  return this->y;  };  };  void main() {  A a, b, c;  a.setx();  a.sety();  b.setx();  b.sety();  c.setx(a.getx() + b.getx());  c.sety(a.gety() + b.gety());  cout << c.getx() << c.gety() << endl;  system("pause");  } | | | | | |
| 问题2.3 | #include "iostream"  using namespace std;  class A {  private:  int x, y;  public:  void setx(int num) { this->x = num; }; //函数重载 有num传入就使用num，没有就从cin获取  void sety(int num) { this->y = num; };  void setx() { cin >> this->x; };  void sety() { cin >> this->y; };  int getx() {  return this->x;  };  int gety() {  return this->y;  };  void add(A a, A b) {  this->x = a.getx() + b.getx();  this->y = a.gety() + b.gety();  };  };  void main() {  A a, b, c;  a.setx();  a.sety();  b.setx();  b.sety();  c.add(a, b);  cout << c.getx() << c.gety() << endl;  system("pause");  } | | | | | |
| 问题2.4 | #include "iostream"  using namespace std;  class A {  private:  int x, y;  public:  A() {  };  void setx(int num) {  cout << "直接使用了" << num << "进行赋值\n"; this->x = num;  cout << "赋值完成\n";  }; //函数重载 有num传入就使用num，没有就从cin获取  void sety(int num) { this->y = num; };  void setx() {  cout << "请从控制台对其进行赋值\n"; cin >> this->x; cout << "赋值完成\n"; };  void sety() {  cout << "请从控制台对其进行赋值\n"; cin >> this->y; cout << "赋值完成\n"; };  int getx() {  cout << "x的值是" << this->x << endl;  return this->x;  };  int gety() {  cout << "y的值是" << this->y << endl;  return this->y;  };  void add(A a, A b) {  cout << "执行加法" << endl;  this->x = a.getx() + b.getx();  this->y = a.gety() + b.gety();  cout << "加法执行完毕 x = " << this->x << " , y = " << this->y << endl;  };  };  void main() {  A a, b, c;  cout << "/////////对对象A/////////" << endl;  a.setx();  a.sety();  cout << "/////////对对象B/////////" << endl;  b.setx();  b.sety();  cout << "/////////对对象C/////////" << endl;  c.add(a, b);  cout << "/////////输出对象C中x和y的值：/////////" << endl;  cout << c.getx() << "\t" << c.gety() << endl;  system("pause");  } | | | | | |
| 程序1（含程序及运行结果截图） | | #include "iostream"  using namespace std;  class A {  public:  int x1, x2;  A() {  cout << "a对象已被创建" << endl;  this->x1 = 1;  this->x2 = 1;  }  ~A() {  cout << "a对象已结束其生命周期" << endl;  }  void run(int n) {  if (n <= 2) while (n-- > 0) cout << x1 << " ";  else {  int temp;  n -= 2;  cout << x1 << " ";  cout << x1 << " ";  while (n-- > 0) {  temp = x1 + x2;  cout << temp << " ";  x1 = x2;  x2 = temp;  }  }  cout << "\n";  };  };  void main() {  A \*a = new A;  a->run(20);  delete a;  system("pause");  } | | | | |
| 程序2（含程序及运行结果截图） | | #include "iostream"  #define ElemType int  #define OK true  #define ERROR false  using namespace std;  class Queue {  public:  Queue(int num); //构建函数，起初始化作用  ~Queue();  ElemType read(); //队首出队  bool write(ElemType num); //队尾入队  bool isEmpty(); //为空判断  bool isFull(); //为满判断  int getLengh(); //获取有效长度  bool extend(int num); //延长队列  private:  int lengh;  int current = 0;  ElemType\* head;  int front = 0;  int tail = 0;  };  bool Queue::isEmpty()  {  if (this->current == 0) return true;  else return false;  }  bool Queue::isFull()  {  if (this->current == this->lengh) return true;  else return false;  }  int Queue::getLengh()  {  return this->current;  }  bool Queue::extend(int num)  {  if (isFull() == false) return false;  else {  ElemType\* temp = this->head;  this->lengh += num;  this->head = new ElemType[this->lengh];  for (int i = 0; i <= current; i++) {  this->head[i] = temp[i];  }  delete[] temp;  return OK;  }  }  Queue::Queue(int num) {  this->lengh = num;  this->head = new ElemType[num];  cout << "初始化成功 环形队列总长度: " << this->lengh << endl;  }  Queue::~Queue() {  cout << "Queue对象已被回收: " << endl;  }  ElemType Queue::read()  {  if (isEmpty() == true) {  cout << "EMPTY!!!" << endl;  return ERROR;  }  else  {  ElemType temp = this->head[this->front];  this->front = (1 + this->front) % this->lengh;  this->current--;  return temp;  }  }  bool Queue::write(ElemType num)  {  if (isFull() == true) {  cout << "FULL!!!" << endl;  return ERROR;  }  else {  this->current++;  this->head[this->tail] = num;  this->tail = (this->tail + 1) % this->lengh;  return OK;  }  };  void main() {  Queue\* a = new Queue (5);  int i = 1;  while (1) {  cin >> i;  if (i) a->write(i);  else {  cout << "以0为结束符，输入完毕" << endl;  break;  }  }  cout << (a->isFull() == true ? "Queue is Full\n" : "Queue is NotFull\n");  cout << "数组的有效长度是：" << a->getLengh() << endl;  while (a->isEmpty() != true) cout << a->read() << " ";  cout << (a->isEmpty() == true ? "Queue is Empty\n" : "Queue is NotEmpty\n");  delete a;  system("pause");  } | | | | |
| 程序3（含程序及运行结果截图） | | #include "iostream"  #define ElemType int  #define OK true  #define ERROR false  using namespace std;  class A {  public:  bool write(ElemType num, int add = 0);  bool extend(int num);  bool init(int num);  bool destory();  void print();  A();  ~A();  private:  int lengh;  int current = 0;  ElemType\* head;  };  void main() {  A\* a = new A;  a->init(20);  a->write(10, 2);  a->print();  a->extend(10);  a->print();  a->write(20);  a->print();  a->extend(-5);  a->print();  a->destory();  delete a;  system("pause");  }  bool A::write(ElemType num, int add)  {  int writeTemp = num;  for (int i = 0; i < this->lengh; i++) {  if (this->head[i] == 0) {  this->head[i] = writeTemp;  writeTemp += add;  }  }  return OK;  }  bool A::extend(int num)  {  ElemType\* lenTemp = this->head;  int temp;  this->head = new ElemType[this->lengh + num];  if (num < 0) temp = this->lengh + num;  else {  temp = this->lengh;  for (int i = this->lengh; i < this->lengh + num; i++)  this->head[i] = 0;  }  for (int i = 0; i < temp; i++)  this->head[i] = lenTemp[i];  delete[] lenTemp;  this->lengh = this->lengh + num;  return OK;  }  bool A::init(int num)  {  this->head = new ElemType[num];  this->lengh = num;  for (int i = 0; i < num; i++)  this->head[i] = 0;  return OK;  }  bool A::destory()  {  delete[] this->head;  this->head = NULL;  return OK;  }  void A::print()  {  for (int i = 0; i < this->lengh; i++)  cout << "No." << i + 1 << " is: " << \*(this->head + i) << endl;  cout << "-----------Print Finished-----------" << endl;  }  A::A() {  cout << "-----------A对象已被建立-----------" << endl;  }  A::~A() {  cout << "-----------A对象已被回收-----------" << endl;  } | | | | |