|  |  |  |
| --- | --- | --- |
| **软件学院实验报告** | | |
| 姓名：胡逢彬 学号：2225060803 专业： 网络工程 年级：2022级 | | |
| 课程名称 | | 数据结构 |
| 实验名称 | | 实验4 顺序栈的基本操作及应用 |
| 实验的准备阶段 | 实验内容 | **实验4、顺序栈的基本操作及应用**  （1）实验目的  通过该实验，让学生掌握栈的相关基本概念，认识栈是插入和删除集中在一端进行的线性结构，掌握栈的“先入后出”操作特点。栈在进行各类操作时，栈底指针固定不动，掌握栈空、栈满的判断条件。  （2）实验内容  用顺序存储结构，实现教材定义的栈的基本操作，提供数制转换功能，将输入的十进制整数转换成二进制、八进制或十六进制。  （3）参考界面  菜单中包括以下功能：  1.初始化栈，2.销毁栈，3.清空栈，4.栈判空，5.求栈长度，6.获取栈顶元素，7.插入一个 元素，8.删除一个元素，9输出所有元素，10进制转换。  要求：自定义的函数中不允许出现提示语和输出语句。  （4）验收/测试用例  通过菜单调用各个操作，测试点：   * 没有初始化前进行其他操作，程序是否能控制住； * 初始化一个栈； * 判栈空，屏幕显示栈为空； * 3个数入栈， 2、4、6； * 栈长度，屏幕输出3； * 取栈顶元素，再判栈空，然后再判栈长度。让学生知道取栈顶元素不改变栈中的内容，栈顶指针不发生改变； * 出栈，再判栈长度和输出栈中内容；（多次出栈，直到栈为空；再出栈，是否提示栈为空） * 销毁栈，再做其他操作，判断程序是否能控制； * 数制转换，（允许用户输入想把十进制转换成几进制），然后灵活的转换成对应的进制。 |
| 实验类型 | 验证性 |
| 实验的重点、难点 | 重点： 入栈和出栈  难点： 进制转换（十六进制） |
| 实验环境 | VC++6.0 |
| 实验的实施阶段 | 实验步骤及完成任务情况 | 一、设计思想  设计思想:  栈是一种线性数据结构，它具有先入后出（LIFO）的特性，即最后进入栈的元素最先被删除。在顺序栈的实现中，我们使用数组来存储栈中的元素。基本操作包括入栈、出栈、判断栈空、判断栈满、获取栈长度等。  基本操作:  1. 初始化栈（InitStack）：在该操作中，我们为栈分配存储空间，并初始化栈顶指针和栈的容量。  2. 销毁栈（DestroyStack）：该操作释放栈所占用的存储空间，并将栈顶指针和栈的容量重置为初始状态。  3. 清空栈（ClearStack）：清空栈即将栈顶指针指向栈底，实际上是清空栈中所有元素，但不释放存储空间。  4. 栈判空（IsEmpty）：判断栈是否为空，即栈顶指针是否等于栈底指针。  5. 求栈长度（StackLength）：通过栈顶指针和栈底指针的差值来求得栈的长度。  6. 获取栈顶元素（GetTop）：通过栈顶指针来获取栈顶元素的值。  7. 插入一个元素（Push）：将元素入栈，将栈顶指针上移，指向新的栈顶元素。  8. 删除一个元素（Pop）：将栈顶元素出栈，将栈顶指针下移，指向新的栈顶元素。  9. 输出所有元素（PrintStack）：输出栈中所有元素的值。  二、主要源代码  #include <iostream>  #define MAX\_SIZE 100  // 定义栈结构  typedef struct {  int\* base; // 栈底指针  int\* top; // 栈顶指针  int stacksize; // 当前已分配的存储空间  } SqStack;  // 初始化栈  void InitStack(SqStack& stack, int capacity) {  stack.base = new int[capacity];  stack.top = stack.base;  stack.stacksize = capacity;  }  // 销毁栈  void DestroyStack(SqStack& stack) {  delete[] stack.base;  stack.top = stack.base;  stack.stacksize = 0;  }  // 清空栈  void ClearStack(SqStack& stack) {  stack.top = stack.base;  }  // 判断栈空  bool IsEmpty(SqStack stack) {  return stack.top == stack.base;  }  // 入栈  bool Push(SqStack& stack, int value) {  \*(stack.top++) = value;  return true;  }  // 出栈  bool Pop(SqStack& stack, int& value) {  if (IsEmpty(stack)) {  return false; // 栈空，出栈失败  }  value = \*(--stack.top);  return true;  }  // 获取栈长度  int StackLength(SqStack stack) {  return stack.top - stack.base;  }  // 获取栈顶元素  bool GetTop(SqStack stack, int& value) {  if (IsEmpty(stack)) {  return false; // 栈空，获取失败  }  value = \*(stack.top - 1);  return true;  }  // 输出栈中所有元素  void PrintStack(SqStack stack) {  int\* k=stack.top;  k--;  int\* p;  for ( p = k; p !=stack.base ; --p) {  if((\*p)>=10)std::cout<<(char)((\*p)+'A'-10)<<" ";  else std::cout << \*p << " ";  }  if((\*p)>=10)std::cout<<(char)((\*p)+'A'-10)<<" ";  else std::cout << \*p << " ";  std::cout << std::endl;  }  // 十进制转二进制  void D10to2(int dec) {  SqStack stack;  InitStack(stack, MAX\_SIZE);  while (dec > 0) {  Push(stack, dec % 2);  dec /= 2;  }  PrintStack(stack);  DestroyStack(stack);  }  // 十进制转八进制  void D10to8(int dec) {  SqStack stack;  InitStack(stack, MAX\_SIZE);    while (dec > 0) {  Push(stack, dec % 8);  dec /= 8;  }  PrintStack(stack);  DestroyStack(stack);  }  // 十进制转十六进制  void D10to16(int dec) {  SqStack stack;  InitStack(stack, MAX\_SIZE);  while (dec > 0) {  int remainder = dec % 16;  Push(stack, remainder);  dec /= 16;  }  PrintStack(stack);  DestroyStack(stack);  }  int main() {  SqStack stack;  int capacity;  std::cout << "请输入栈的最大容量：";  std::cin >> capacity;  InitStack(stack, capacity);  int choice;  int value;  while (true) {  // 显示菜单省略，与上一个代码示例相同  system("pause");  system("cls");  std::cout << "菜单：" << std::endl;  std::cout << "1. 初始化栈" << std::endl;  std::cout << "2. 销毁栈" << std::endl;  std::cout << "3. 清空栈" << std::endl;  std::cout << "4. 栈判空" << std::endl;  std::cout << "5. 求栈长度" << std::endl;  std::cout << "6. 获取栈顶元素" << std::endl;  std::cout << "7. 插入一个元素" << std::endl;  std::cout << "8. 删除一个元素" << std::endl;  std::cout << "9. 输出所有元素" << std::endl;  std::cout << "10. 进制转换" << std::endl;  std::cout << "0. 退出" << std::endl;  std::cout << "请输入功能对应的数字：";  std::cin >> choice;    switch (choice) {  case 1:  InitStack(stack,MAX\_SIZE);  std::cout << "栈已初始化！" << std::endl;  break;  case 2:  DestroyStack(stack);  std::cout << "栈已销毁！" << std::endl;  break;  case 3:  ClearStack(stack);  std::cout << "栈已清空！" << std::endl;  break;  case 4:  if (IsEmpty(stack)) {  std::cout << "栈为空！" << std::endl;  } else {  std::cout << "栈不为空！" << std::endl;  }  break;  case 5:  std::cout << "栈长度为：" << StackLength(stack) << std::endl;  break;  case 6:  if (GetTop(stack, value)) {  std::cout << "栈顶元素为：" << value << std::endl;  } else {  std::cout << "栈为空，没有栈顶元素！" << std::endl;  }  break;  case 7:  std::cout << "请输入要插入的元素：";  std::cin >> value;  if (Push(stack, value)) {  std::cout << "插入成功！" << std::endl;  } else {  std::cout << "插入失败，栈已满！" << std::endl;  }  break;  case 8:  if (Pop(stack, value)) {  std::cout << "删除的元素为：" << value << std::endl;  } else {  std::cout << "删除失败，栈为空！" << std::endl;  }  break;  case 9:  std::cout << "栈中所有元素为：";  PrintStack(stack);  break;  case 10:  std::cout << "请输入要转换的十进制数：";  std::cin >> value;  std::cout << "二进制表示：";  D10to2(value);  std::cout << "八进制表示：";  D10to8(value);  std::cout << "十六进制表示：";  D10to16(value);  break;  case 0:  std::cout << "程序已退出！" << std::endl;  DestroyStack(stack);  return 0;  default:  std::cout << "输入无效，请重新输入！" << std::endl;  break;  }  }  return 0;  } |
| 实验结果的处理阶段 | 实验结果 | IMG_256  IMG_256  IMG_256  IMG_256  IMG_256  IMG_256  IMG_256 |
| 实验结果总结 |  |