# 目录

[1 1](#_Toc161837931)

[2 4](#_Toc161837932)

[3 8](#_Toc161837933)

[4 11](#_Toc161837934)

# 1

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  class myString  {  public:      myString(const char \*pn = NULL)      {          if (pn == NULL)          {              pStr = new char[1];              pStr[0] = '\0';              size = 0;          }          else          {              size = strlen(pn);              pStr = new char[size + 1];              strcpy(pStr, pn);          }      }      myString(const myString &other)      {          if (other.pStr == NULL)          {              pStr = new char[1];              pStr[0] = '\0';              size = 0;          }          else          {              size = other.size;              pStr = new char[size + 1];              strcpy(pStr, other.pStr);          }      }      ~myString()      {          delete[] pStr;      }      void set(const char \*pn)      {          if (pStr != NULL)          {              delete[] pStr;          }          if (pn == NULL)          {              pStr = new char[1];              pStr[0] = '\0';              size = 0;          }          else          {              size = strlen(pn);              pStr = new char[size + 1];              strcpy(pStr, pn);          }      }      void print()      {          if (size == 0)              cout << "空字符串" << endl;          else              cout << pStr << endl;      }      void toUpper()      {          for (int i = 0; i < size; i++)          {              if (pStr[i] >= 'a' && pStr[i] <= 'z')              {                  pStr[i] += 'A' - 'a';              }          }      }      void toLower()      {          for (int i = 0; i < size; i++)          {              if (pStr[i] >= 'A' && pStr[i] <= 'Z')              {                  pStr[i] += 'a' - 'A';              }          }      }      int getLen()      {          return size;      }  private:      char \*pStr;      int size;  };  int main()  { // 测试类      myString s0, s1("hello");      s0.print(); // 输出“空字符串”      s0.set("world");      s0.print(); // 输出“world”      s1.toUpper();      cout << s1.getLen() << endl; // 输出 5      myString s2 = s1;      s2.print(); // 输出“HELLO”      s2.toLower();      s1.print(); // 输出?      return 0;  } |
|  |

# 2

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  int isRun(int year)  {      if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))          return 1;      else          return 0;  }  class Date  {  public:      Date(int y, int m, int d)      {          if (is\_valid(y, m, d))          {              year = y;              month = m;              day = d;          }          else          {              year = 2023;              month = 8;              day = 28;          }      }      Date()      {          year = 2023;          month = 8;          day = 28;      }      bool set(int y, int m, int d)      {          year = y;          month = m;          day = d;          return true;      }      void print()      {          cout << month << "/" << day << "/" << year << endl;      }      int day\_of\_year()      {          int days[] = {0, 31, 28 + isRun(year), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};          int i = 0;          int DiJiTiannn = 0;          for (i = 1; i < month; i++)          {              DiJiTiannn += days[i];          }          DiJiTiannn += day;          return DiJiTiannn;      }      int diff(const Date &A)      {          int First = Date::day\_of\_year();          int days[] = {0, 31, 28 + isRun(A.year), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};          int i = 0;          int DiJiTiannn = 0;          for (i = 1; i < A.month; i++)          {              DiJiTiannn += days[i];          }          DiJiTiannn += A.day;          int Second = DiJiTiannn;          for (int i = 1999; i < year; i++)          {              First += 365;              First += isRun(i);          }          for (int i = 1999; i < A.year; i++)          {              Second += 365;              Second += isRun(i);          }          return First - Second;      }  private:      bool is\_valid(int y, int m, int d)      {          if (y < 1 || m < 1 || m > 12 || d < 1)              return false;          int days[] = {0, 31, 28 + isRun(y), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};          if (d > days[m])              return false;          return true;      }      int year;      int month;      int day;  };  class Student  {  private:      int id;      char name[20];      Date roll;      static int number;  public:      Student(const char \*ss = "ssdut", int yyear = 2023, int mmonth = 8, int dday = 23)      {          roll.set(yyear, mmonth, dday);          if (ss != NULL)          {              strcpy(name, ss);          }          else          {              strcpy(name, "\0");          }          id = 2023000 + (++number);      }      void show()      {          cout << "学号: " << id << endl;          cout << "名字: " << name << endl;          cout << "入学日期: ";          roll.print();      }      static int getNumber()      {          return number;      }  };  int Student::number = 0;  int main()  {      Student s[4] = {Student("S1", 1994, 8, 3), Student("S2", 2022), Student("刘晨旭")};      for (int i = 0; i < 4; i++)      {          cout << "第" << i + 1 << "个学生： " << endl;          s[i].show();          cout << "static int number =";          cout << s[i].getNumber();          cout << endl               << endl;      }      return 0;  } |
|  |

# 3

|  |
| --- |
| #include <iostream>  using namespace std;  typedef long long ll;  class Stack  {  public:      Stack(int n = 0);  // 初始化, 栈的容量为 n      ~Stack();          // 析构函数      bool push(char x); // 将 x 入栈存在栈顶位置，返回知否入栈成功（如果栈已满则      char pop();   // 将栈顶元素出栈，返回栈顶元素，并修改栈顶位置      bool empty(); // 判断栈是否为空      bool full();  // 判断栈是否已满  private:      char \*buffer;      int top;  // 栈顶位置      int size; // 栈的容量(最多能存放的元素个数)  };  Stack::Stack(int n)  {      buffer = new char[n];      size = n;      top = 0;  }  Stack::~Stack()  {      delete[] buffer;  }  bool Stack::push(char x)  {      if (top < size)      {          buffer[top++] = x;          return 1;      }      else          return 0;  }  char Stack::pop()  {      if (top == 0)      {          return 0;      }      else      {          return buffer[--top];      }  }  bool Stack::empty()  {      if (top == 0)          return 1;      else          return 0;  }  bool Stack::full()  {      if (top == size)          return 1;      else          return 0;  }  int main()  {      ios::sync\_with\_stdio(false);      cin.tie(0);      cout.tie(0);        Stack a(10);      for (int i = 1; i <= 10; i++)      {          a.push(i + 'a' - 1);      }      if (a.empty())          cout << "it is empty" << endl;      else          cout << "not empty" << endl;      if (a.full())          cout << "it is full" << endl;      else          cout << "it is not full" << endl;      for (int i = 1; i <= 10; i++)      {          cout << a.pop() << endl;      }      if (a.empty())          cout << "it is empty" << endl;      else          cout << "not empty" << endl;      if (a.full())          cout << "it is full" << endl;      else          cout << "it is not full" << endl;      return 0;  } |
|  |

# 4

|  |
| --- |
| #include <iostream>  using namespace std;  typedef long long ll;  class Point  {  private:      double x;      double y;  public:      Point(Point &p)      {          x = p.x;          y = p.y;      }      Point(double xx = 0, double yy = 0)      {          x = xx;          y = yy;      }      void set(double xx, double yy)      {          x = xx;          y = yy;      }      void show()      {          cout << x << ' ' << y << endl;      }  };  class PointSet  {  private:      int num;        // 点的实际数量      Point point[5]; // Array of points  public:      PointSet(int n = 0); // 初始没有点      bool add(Point &p);  // 加入一个点.      // 如果超过 5 个点则加入失败.      Point get(int n) // 获得第 n 号点, 点的排序从 0 号开始      {          return point[n - 1];      }      // 检查 n 的合理性. 如果该点不存在，则显示提示信息.      void setAll(double x, double y);            // 将所有的点设置为同一个坐标 x, y      int getNum();                               // 返回集合中点的个数      bool resetPoint(int n, double x, double y); // 将第 n 号点的坐标设置为 x，y      // 如果该点不存在 return false      void show(); // 输出所有的点  };  PointSet::PointSet(int n)  {      num = n;  }  bool PointSet::add(Point &p)  {      if (num >= 5)          return false;      else      {          point[num++] = p;          return true;      }  }  void PointSet::setAll(double xx, double yy)  {      for (int i = 0; i < num; i++)      {          point[i].set(xx, yy);      }  }  int PointSet::getNum()  {      return num;  }  bool PointSet::resetPoint(int n, double xx, double yy)  {      if (n > num || n <= 0)          return false;      else      {          point[n - 1].set(xx, yy);          return true;      }  }  void PointSet::show()  {      for (int i = 0; i < num; i++)      {          cout << "第" << i + 1 << "个";          point[i].show();      }      cout << endl;  }  int main()  {      PointSet a(2);      a.show();      Point p(2, 2);      a.add(p);      a.show();      p.set(2.2, 3.3);      a.add(p);      a.show();      cout << "里面现在有" << a.getNum() << "个point" << endl;      a.resetPoint(4, 1145.14, 22.33);      a.show();      a.add(p);      a.show();      p.set(123, 321);      a.add(p);      a.show();      a.get(4).show();      a.setAll(1, 1);      a.show();      return 0;  } |
|  |