**Các bài thực hành môn Lập Trình**

1. **Đối tượng**
   1. Sinh viên đã có kiến thức cơ bản về lập trình C
2. **Mục đích bài thực hành**
   1. Giúp sinh viên hiểu và vận dụng được các kiến thức cơ bản về lập trình trong C, C++ và các kiến thức cơ bản về giải thuật và cấu trúc dữ liệu
   2. Giúp sinh viên có được kỹ năng lập trình để có thể tự mình viết được các chương trình giải quyết các bài toán từ đơn giản đến mức trung bình mà được nêu ra trong môn học
3. **Đánh giá**
   1. Sau mỗi buổi thực hành, sinh viên cần hoàn thành bài thực hành và nộp lại cho giáo viên
   2. Bài thực hành sẽ có nhiều câu với nhiều mức độ khác nhau, có thang điểm cụ thể cho từng câu
   3. Nội dung thực hành sẽ đưa trước để sinh viên có thể chuẩn bị trước ở nhà, nhưng các tham số test có thể sẽ phát riêng cho mỗi sinh viên khi vào làm bài
   4. Sinh viên sẽ phải tự edit bài thực hành của mình trong thời gian làm bài.
   5. Giáo viên hướng dẫn sẽ có nhiệm vụ hướng dẫn cho sinh viên làm bài khi gặp các khó khăn như gặp lỗi, lỗi dịch, lỗi chạy, lỗi công cụ lập trình, lỗi hệ thống,v.v
4. **Nội dung các bài thực hành**

Gồm có 3 bài thực hành, mỗi bài có thời lượng 2h

**Bài 0**

Hướng dẫn sử dụng công cụ lập trình

Trong bài này sinh viên sẽ làm quen với công cụ lập trình **Dev-C++** (phiên bản 4.9.9)

Một số ưu điểm của công cụ này:

* Môi trường lập trình trong Windows cho cả C và C++ miễn phí, dung lượng khá nhỏ (chỉ khoảng 10Mb), yêu cầu cấu hình máy tính rất đơn giản (từ Win95 trở lên, chỉ cần 32M RAM)
* Có hỗ trợ soạn thảo và chạy chương trình ở cả 2 chế độ single file và project.

Một số chức năng cơ bản của công cụ:

1. Tạo một file nguồn mới để bắt đầu soạn thảo chương trình (File🡪New🡪Source File)

Hướng dẫn chi tiết tạo một file nguồn để viết chương trình Hello World

1. Tạo một project mới để quản lý nhiều file nguồn (File🡪New🡪Project)

Hướng dẫn chi tiết tạo một project để viết một chương trình tính diện tích một hình tròn, gồm có 2 tệp. Một tệp chứa hàm main, còn tệp kia chứa các hàm nhập dữ liệu và tính diện tích.

**Bài 1**

Bài này giúp sinh viên làm quen với các kiểu dữ liệu cơ bản, và bước đầu với kiểu mảng 1 chiều, cũng như một số thao tác cơ bản như nhập, xuất, gán.

**Program 1.1**: Viết 1 chương trình thực hiện các công việc sau:

1. Nhập một kí tự từ bàn phím
2. In kí tự đó và mã ASCII, mã cơ số 8 và cơ số 16 của nó ra màn hình

**Program 1.2**: Viết chương trình tính điện kháng của một tụ điện theo công thức: Xc = 1/(2πfC), với f và C tương ứng là tần số và điện dung của tụ. Chương trình sẽ thực hiện các công việc sau:

1. Tự khai báo hằng số π.
2. Nhập các giá trị f, C từ bàn phím
3. Tính điện kháng rồi in ra màn hình

**Program 1.3**: Viết chương trình tính điện trở tương đương của 3 điện trở mắc nối tiếp hoặc song song. Chương trình sẽ thực hiện các công việc sau:

1. Nhập giá trị từng điện trở từ bàn phím
2. Nhập cách mắc các điện trở (nối tiếp hoặc song song)
3. Tính và in kết quả ra màn hình.

**Bài 2**

Cho phép sinh viên làm quen với các cấu trúc điều khiển cơ bản. Đồng thời thao tác với mảng nhiều chiều.

**Program 2.1**: Mở rộng Program 1.3, viết chương trình tính điện trở tương đương của N điện trở mắc nối tiếp hoặc song song. Chương trình sẽ thực hiện các công việc sau:

1. Nhập số điện trở N và giá trị từng điện trở từ bàn phím. Có kiểm tra tính hợp lệ của dữ liệu nhập vào (giá trị N>0 và <20; các giá trị điện trở phải >0).
2. Nhập cách mắc các điện trở (nối tiếp hoặc song song)
3. Tính và in kết quả ra màn hình.

**Program 2.2**: Viết chương trình cho phép đếm số từ có trong một chuỗi. Yêu cầu:

1. Nhập một chuỗi có độ dài không quá 200 kí tự. Có kiểm tra tính hợp lệ của dữ liệu nhập vào.
2. Đếm số từ có trong chuỗi đó và in kết quả ra màn hình.

**Program 2.3**: Viết chương trình thực hiện các yêu cầu sau:

1. Nhập một ma trận kích thước MxN kiểu số thực (0<M,N<10), và các giá trị của các phần tử trong đó từ bàn phím.
2. Tính tổng các phần tử theo từng hàng, và theo từng cột rồi in ra. In cả tổng của toàn bộ các phần tử của ma trận.

**Bài 3**

Các bài tập cơ bản về C++, như tạo class, định nghĩa các hàm thành viên, các hàm tạo và hàm hủy, định nghĩa lại các phép toán

**Program 3.1.** Cho lớp mô tả số phức sau:

|  |
| --- |
| **class Complex** |
| class Complex  {  private:  double re, im;  public:  // Hàm tạo thiết lập từ 2 tham số  Complex(double r = 0, double i = 0) : re(r), im(i) { }  // Hàm tạo copy từ một số phức  Complex(Complex& c) : re(c.re), im(c.im) { }  public:  Complex operator + (Complex c); // Hàm định nghĩa toán tử cộng 2 số phức  Complex operator - (Complex c); // Hàm định nghĩa toán tử trừ 2 số phức  Complex operator \* (Complex c); // Hàm định nghĩa toán tử nhân 2 số phức  Complex operator / (Complex c); // Hàm định nghĩa toán tử chia 2 số phức  public:  // Hàm định nghĩa toán tử luồng ra  friend ostream& operator << (ostream& out, Complex& c)  {  return (out << '(' << c.re << ", " << c.im << "i)");  }  };  Complex Complex::operator + (Complex c)  {  return Complex(this->re + c.re, this->im + c.im);  } |

1. Viết code cho các toán tử còn lại của lớp Complex
2. Kiểm tra kết quả bằng chương trình sau đây:

|  |
| --- |
| void main()  {  // Khai báo 2 số phức y và z, mỗi số được khởi tạo từ 1 hoặc 2 số thực bất kỳ  // Khai báo biến double a và khởi tạo bằng một số thực bất kỳ  cout << y << " + " << z << " = " << y + z + a << end;  cout << y << " - " << z << " = " << y - z << end;  cout << y << " \* " << z << " = " << y \* z << end;  cout << y << " / " << z << " = " << y / z << end;  system("pause");  } |

1. Báo cáo thực hành:
   * Các đoạn code cho các toán tử
   * Đoạn khai báo các biến
   * Kết quả trên màn hình

**Program 3.2.** Cho lớp mô tả xâu ký tự sau:

|  |
| --- |
| **class String** |
| class String  {  private:  int length;  char \* data;  private:  // Hàm tạo String từ mảng ký tự với độ dài cho trước  String(int length, char \*data) : length(length), data(data) { }  public:    // Hàm tạo mặc định tạo String rỗng  String() : length(0), data(new char[1]) { data[0] = 0; }  // Hàm tạo String từ một hằng xâu ký tự  String(const char\* s)  {  length = strlen(s);  data = new char[length + 1];  strcpy(data, s);  }  // Hàm tạo String copy từ một String đã có  String(String& s);  // Hàm hủy  ~String() { delete[] data; }  public:  // Hàm so sánh hai String  // Giá trị trả về -1, 0, 1 tương ứng String <, ==, > String s  int Compare(String s);    public:  // Toán tử chỉ số truy cập ký tự tại chỉ số index  char & operator[] (int index) { return data[index]; }  // Toán tử gán s vào String  String& operator = (String& s);  public:  // Toán tử cộng thực hiện thêm ký tự c vào cuối String  String operator + (char c);  // Toán tử cộng nối s vào cuối String  String operator + (String s);  public:  // Toán tử so sánh bằng  int operator == (String s);  // Toán tử so sanh khác  int operator != (String s);  // Toán tử luồng ra in String ra màn hình  friend ostream& operator << (ostream& out, String& s)  {  return (out << s.data);  }  };  String & String::operator = (String& s)  {  delete[] data;  length = s.length;  data = new char[length + 1];  strcpy(data, s.data);  return (\*this);  }  String String::operator + (char c)  {  int length = this->length + 1;  char \* data = new char[length + 1];  strcpy(data, this->data);  data[this->length] = c;  data[length] = 0;  return String(length, data);  } |

1. Viết code cho các hàm và các toán tử còn lại của lớp String
2. Kiểm tra kết quả bằng chương trình sau đây:

|  |
| --- |
| void main()  {  // Khai báo một String s và khởi tạo từ một hằng xâu ký tự bất kỳ  // Viết biểu thức in ra màn hình khi thêm một ký tự bất kỳ vào s  // Viết biểu thức in ra màn hình khi nối một hằng xâu ký tự bất kỳ vào s  system("pause");  } |

1. Báo cáo thực hành:
   * Các đoạn code cho các hàm và các toán tử
   * Các biểu thức của hàm main
   * Kết quả trên màn hình