Nguyễn Văn Quân

**Lập trình C cơ bản**

**Cachhoc.net**

**Mục lục:**

Bài 1 – Hello World 2

Bài 2 – Các thành phần trong C 6

Bài 3 – Nhập xuất trong C 11

Bài 4 – Phép toán, toán tử trong C 17

Bài 5 – if else, switch case trong C 23

Bài 6 – Vòng lặp for trong C 31

Bài 7 – Vòng lặp while, do…while trong C 37

Bài 8 – Hàm trong C 41

Bài 9 – Mảng trong C 47

Bài 10 – Chuỗi ký tự trong C 53

Bài 11 – Con trỏ trong C 58

Bài 12 – Mối quan hệ giữa con trỏ và mảng, chuỗi ký tự 63

Bài 13 – Kiểu cấu trúc – struct 70

Bài 14 – Nhập xuất file trong C/C++ 78

**Lập trình C: Bài 1 – Hello World**

**Nội dung**

* [1. Giới thiệu về lập trình](https://cachhoc.net/2014/12/01/bai-1-hello-world/#1_Gioi_thieu_ve_lap_trinh)
* [2. Lập trình C](https://cachhoc.net/2014/12/01/bai-1-hello-world/#2_Lap_trinh_C)
* [3. Cài đặt công cụ](https://cachhoc.net/2014/12/01/bai-1-hello-world/#3_Cai_dat_cong_cu)
  + [3.1 Cài đặt](https://cachhoc.net/2014/12/01/bai-1-hello-world/#31_Cai_dat)
  + [3.2 Giới thiệu cơ bản về Dev-C++](https://cachhoc.net/2014/12/01/bai-1-hello-world/#32_Gioi_thieu_co_ban_ve_Dev-C)
* [4. Tạo chương trình đầu tiên](https://cachhoc.net/2014/12/01/bai-1-hello-world/#4_Tao_chuong_trinh_dau_tien)

**1. Giới thiệu về lập trình**

Chắc hẳn bạn đã được tiếp xúc với máy tính, điện thoại dù ít hay nhiều và dùng các ứng dụng hoặc trò chơi trên đó ví như Word, Excel để làm việc văn phòng hoặc dùng Chrome, Firefox để vào Facebook, Google hay chính Facebook và Google là các phầm mềm.

Bạn tưởng tượng lập trình là quá trình bạn tạo ra các phần mềm, ứng dụng như thế. Bạn tạo ra ứng dụng cho người khác dùng, làm việc; bạn tạo ra game cho người khác chơi, làm ra website cho người khác truy cập vào. Nghĩ đến đây đã thấy hứng thú rồi

Để làm ra được phần mềm (hoặc game, website) thì chúng ta phải dạy cho máy tính cách làm việc hay nói cách khác là đưa các quy tắc mà chúng ta mong muốn để máy tính thực hiện như là khi ấn nút lên trên thì nhân vật phải nhảy lên, ấn nút sang trái thì nhân vật sẽ đi sang trái.

Giống như dạy cho ai đó phải làm gì thì bạn phải nói cho họ nghe hoặc viết chữ để đưa cho họ đọc và từ đó họ mới hiểu được bạn nói gì. Việc đưa các quy tắc này vào máy tính để nó hiểu được bạn muốn làm gì mà thực hiện theo cũng vậy, bạn cần viết các yêu cầu đó theo một cú pháp cụ thể mà cái đó gọi là **ngôn ngữ lập trình**.

**Ngôn ngữ lập trình** cũng giống như tiếng Việt để người Việt giao tiếp và hiểu, tiếng Anh dùng cho người Anh, Mỹ và nhiều nước khác, … **ngôn ngữ lập trình** nó là ngôn ngữ để người nói cho máy tính hiểu. Hiện nay có nhiều ngôn ngữ lập trình khác nhau như C, C++, Java, C#, PHP, python,… mỗi ngôn ngữ có ưu nhược điểm riêng và có ứng dụng trong các lĩnh vực nhất định.

**2. Lập trình C**

Lập trình C đơn giản là các bạn dùng **Ngôn ngữ lập trình C** để tạo ra các phần mềm.

Mỗi phần mềm hay gọi là chương trình (chương trình máy tính) sẽ được viết bởi ngôn ngữ C bằng một hay nhiền file với những quy tắc nhất định của ngôn ngữ mà chúng ta sẽ dần tìm hiểu.

**3. Cài đặt công cụ**

**3.1 Cài đặt**

Để có thể lập trình C thì các bạn cần cài bộ dịch gcc hoặc g++. Tuy nhiên để dễ dàng thì hiện tại có nhiều phần mềm tích hợp (hay gọi là môi trường phát triển – IDE) để chúng ta có thể dễ dàng.

* Đối với các bạn dùng Windows có thể download [Dev-C++](https://sourceforge.net/projects/orwelldevcpp/) hoặc [Code::Blocks](http://www.codeblocks.org/downloads/binaries) về cài đặt và chúng ta có thể dùng ngay để viết các chương trình đơn giản.
* Đối với các bạn dùng Linux (ubuntu, Linux Mint,…) có thể cài bằng cách mở terminal lên và chạy lệnh cài đặt sau:

sudo apt-get install build-essential

Sau khi chạy lệnh trên, các bạn có thể mở bất cứ trình soạn thảo nào để code (như gedit, vim, hay sublime-text,…) sau đó dịch, chạy chương trình bằng terminal.

Hoặc các bạn cũng hoàn toàn có thể download và cài [Code::Blocks](http://www.codeblocks.org/downloads/binaries) trên Linux để dùng rất tiện, hoặc dùng lệnh cài:

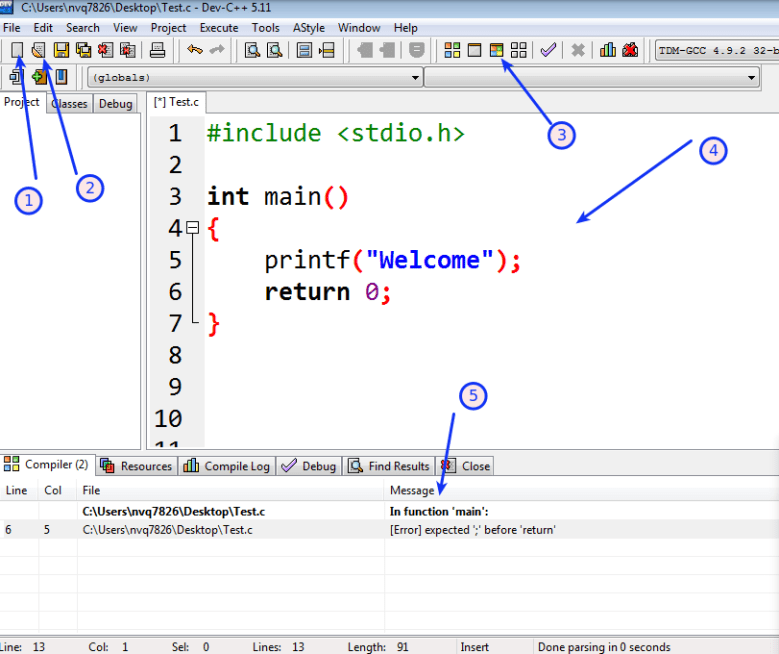
sudo apt-get install codeblocks

Các bạn có thể xem thêm chi tiết về [Lập trình C/C++ trên Ubuntu (Linux) tại blog của mình](https://nguyenvanquan7826.com/2014/05/21/linux-cc-lap-trinh-cc-tren-ubuntu-linux/)

Để dễ và thống nhất trong quá trình làm, mình sẽ dùng Dev-C++ nhé.

**3.2 Giới thiệu cơ bản về Dev-C++**

Để tiện hình dung, các bạn mở Dev-C++ lên sẽ thấy giao diện như hình sau (*Hiện tại bản mình đang dùng là 5.11, các bạn dùng bản khác có thể sẽ có một chút khác nhưng sẽ không đáng kể)*.



*Giao diện Dev-C++ 5.11*

Trong hình trên, có một số phần chính mình đánh dấu bằng các số với ý nghĩa sau:

1. Nút tạo file mới

2. Nút mở một file đã có

3. Nút biên dịch và chạy chương trình

4. Vùng soạn thảo code (mã chương trình)

5. Vùng hiển thị lỗi nếu có.

**4. Tạo chương trình đầu tiên**

Bây giờ các bạn có thể viết chương trình phần mềm đầu tiên để thử nghiệm.

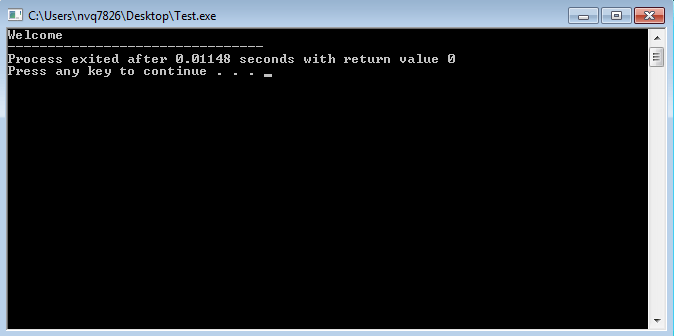
B1: Tạo 1 file mới.

B2: Gõ hoặc chép đoạn code sau vào vùng soạn thảo.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include <stdio.h>    int main()  {      printf("Welcome");      return 0;  } |

B3: Lưu lại với tên file là **Test.c** hoặc **Test.cpp**

B4: Click nút chạy chương trình và bạn sẽ thấy một màn hình màu đen hiện lên như sau:



*Chương trình C đầu tiên khi chạy*

Trong đó các bạn có thể thấy chữ ***Welcome*** hiện lên, đó chính là dòng chữ được in ra khi dùng lệnh **printf**. Phần dưới đường kẻ ngang là do Dev-C tự sinh ra.

**Mình giải thích một chút về code:**

* **Dòng 1:** Chứa phát biểu tiền xử lý #include <stdio.h> . Vì trong chương trình này ta sử dụng các lệnh trong thư viện của C là printf, do đó bạn cần phải có khai báo của hàm thư viện này để báo cho trình biên dịch C biết. Nếu không khai báo chương trình sẽ báo lỗi. Thư viện **stdio.h** viết tắt của **standard input output** (std – i – o) là thư viện nhập xuất chuẩn).
* **Dòng 3: \*\* int main() là thành phần chính của mọi chương trình C. Mọi chương trình C đều bắt đầu thi hành từ hàm \*\*main**. Cặp dấu ngoặc () cho biết đây là khối hàm (function). Hàm main() có từ khóa int đầu tiên cho biết hàm này trả về giá trị kiểu nguyên (int).
* **Dòng 4 và 7:** cặp dấu ngoặc móc {} giới hạn thân của hàm. Thân hàm bắt đầu bằng dấu { và kết thúc bằng dấu }.
* **Dòng 5:** printf ("Welcome");, chỉ thị cho máy in ra chuỗi ký tự nằm trong nháy kép "". Hàng này được gọi là một câu lệnh, kết thúc một câu lệnh trong C phải là dấu chấm phẩy( ; ).
* **Dòng 6:** return 0; Trả về giá trị kiểu nguyên là 0 theo như đúng ban đầu là khai báo int main().

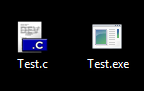
**Lưu ý:**

1. Trong chương trình này mình **không dùng thư viện conio.h** vì trong chuẩn C không có thư viện này, và từ đó cũng không dùng được getch() để dừng màn hình mà mình đã thay bằng lệnh system("pause"); trong thư viện **stdlib.h**
2. Khi dùng return để trả về giá trị của hàm thì các bạn có thể bỏ qua lệnh này chương trình vẫn chạy nhưng về chuẩn là sai, trả về 1 cũng sai, tóm lại là trả về 0. Nếu trả về 1 số khác không thì hệ thống máy tính sẽ hiểu là chương trình này của bạn sau khi chạy phát sinh ra cần đó lỗi.

Bây giờ các bạn mở folder chứa file **Test.c** mà bạn vừa tạo ra, các bạn sẽ nhìn thấy một file có tên **Test.exe** (gọi tắt là file exe), đây chính là file chạy chương trình. Bây giờ nếu muốn bạn có thể click chuột vào file exe này là chạy được chương trình trên hoặc bạn có thể copy file exe này sang máy tính khác thì vẫn có thể chạy được, nó giống như bạn copy game từ máy này sang máy khác vậy.

Từ đây ta có:

* File **Test.c** là file chúng ta tạo ra và viết các lệnh để máy tính hiểu, file này gọi là **file mã nguồn**.
* File **Test.exe** là file sinh ra khi chúng ta ấn nút Chạy chương trình, file này gọi là **file thực thi**.



*File mã nguồn và file thực thi*

***Lưu ý: Các chương trình sau này chúng ta làm sẽ chủ yếu chạy trên màn hình đen như trên (gọi là màn hình console – đọc là ‘con sâu’), tuy không có giao diện đẹp mắt nhưng đây là phần lõi của lập trình, học tốt phần này chúng ta mới có thể làm được các chương trình có giao diện đồ họa và các ngôn ngữ khác cũng hầu hết dựa vào phần lõi này.***

**Lập trình C: Bài 2 – Các thành phần trong C**

**Nội dung**

* [1. Các ký tự điều khiển](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#1_Cac_ky_tu_dieu_khien)
* [2. Bộ chữ viết](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#2_Bo_chu_viet)
* [3. Từ khóa](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#3_Tu_khoa)
* [4. Kiểu dữ liệu](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#4_Kieu_du_lieu)
* [5. Tên](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#5_Ten)
* [6. Biến](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#6_Bien)
  + [6.1 Khai báo biến](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#61_Khai_bao_bien)
  + [6.2 Phạm vi của biến](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#62_Pham_vi_cua_bien)
* [7. Hằng](https://cachhoc.net/2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#7_Hang)

**1. Các ký tự điều khiển**

* \n : Nhảy xuống dòng kế tiếp canh về cột đầu tiên.
* \t : Tab ngang.
* \r : Nhảy về đầu hàng, không xuống hàng.
* \a : Tiếng kêu bip.
* \\ : In ra dấu \
* \” : In ra dấu “
* \’ : In ra dấu ‘
* %%: In ra dấu %

Đây chỉ là một số ký tự điểu khiển quen thuộc, hay dùng, ngoài ra còn một só ký tự điều khiển khác các bạn có thể xem thêm trong các tài liệu khác. Để hiểu rõ về các ký tự điều khiển các bạn hãy chạy thử chương trình sau và tự rút ra nhận xét cho riêng mình.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19 | #include <stdio.h>    int main()  {      printf("\a");      printf("Hinh nhu vua co tieng gi keu @@\n");      printf("Ban dang o dong thu 2\nBay gio xuong dong 3 roi ne ^^\n");      printf("\tDong nay cach ra 1 tab thi phai?\n");      printf("\t\t\t\t\t\tCach ra nhieu tab qua \rVe dau dong thoi\n");      printf("Dau \\ \nDau \'\nDau \" \nDau %%\n");        // day la mot dong ghi chu va chi co tac dung chu thich khi viet code cho de hieu        /\*        Day la mot doan ghi chu        Doan ghi chu nay co 2 dong       \*/      return 0;  } |

**2. Bộ chữ viết**

Bộ chữ viết trong ngôn ngữ C bao gồm những ký tự, ký hiệu sau: (phân biệt chữ in hoa và in thường):

* 26 chữ cái latinh hoa A,B,C…Z
* 26 chữ cái latinh thường a,b,c …z.
* 10 chữ số thập phân 0,1,2…9.
* Các ký hiệu toán học: +, -, \*, /, =, <, >, (, )
* Các ký hiệu đặc biệt: :. , ; ” ‘ \_ @ # $ ! ^ [ ] { } …
* Dấu cách hay khoảng trống, xuống hàng (\n) và tab (\t)

**3. Từ khóa**

Là các từ mà ngôn ngữ C đã xây dựng sẵn và có những chức năng nhất định, sau này làm nhiều các bạn sẽ biết dần dần. Chúng ta không nên định nghĩa lại những từ khóa này.

***asm • auto • break • case • cdecl • char • class • const • continue • \_cs • default • delete • do double • \_ds • else • enum • \_es • extern • \_export • far • \_fastcall • float • for •friend • goto • huge • if • inline • int • interrupt • \_loadds • long • near • new • operator • pascal • private • protected • public • register • return • \_saveregs • \_seg • short •signed • sizeof • \_ss • static • struct • switch • template • this • typedef • union • unsigned • virtual • void • volatile • while***

**4. Kiểu dữ liệu**

Trong cuộc sống có nhiều loại vật dụng để chứa đựng các thứ khác nhau như Rổ để đựng rau, Cốc (ca) để đựng nước. Cũng giống vậy, trong lập trình có các kiểu dữ liệu để có thể biểu diễn (lưu trữ) các thành phần tương ứng.

Mỗi kiểu dữ liệu sẽ có kích thước (chiếm diện tích), loại giá trị và miền giá trị khác nhau trong bộ nhớ máy tính khi sử dụng. Dưới đây là một số kiểu dữ liệu cơ bản hay dùng trong C.

| **Loại dữ liệu** | **Kiểu dữ liệu** | **Kích thước (byte)** | **Miền giá trị** |
| --- | --- | --- | --- |
| Kiểu Ký tự | char | 1 | -128 đến 127 |
| unsigned char | 1 | 0 đến 255 |
| Kiểu Số nguyên | short | 2 | -32768 đến 32767 |
| int | 4 | -2147483648 đến 2147483647 |
| long | 4 | -2147483648 đến 2147483647 |
| long long | 8 | -9223372036854775808 đến 9223372036854775807 |
| Kiểu Số thực | float | 4 | 1.175494-38 đến 3.40282338 |
| double | 8 | 2.225074308 đến 1.797693308 |
| long double | 12 | 3.362103-4932 đến 1.1897314932 |

**Lưu ý:**

* Trên đây là bảng thông tin của một số kiểu dữ liệu thông dụng hay dùng. Còn khá nhiều kiểu dữ liệu khác nhưng tôi không đưa vào, các bạn có thể tham khảo tại [đây cho Kiểu số nguyên](http://www.cplusplus.com/reference/climits/), [đây cho kiểu số thực](http://www.cplusplus.com/reference/cfloat/)
* Do C (hoặc C++) có nhiều phiên bản khác nhau và dùng trên các hệ điều hành khác nhau nên có thể giá trị và kích thức các kiểu dữ liệu cũng khác nhau. Bảng trên là tôi kiểm tra trên *win 7, Dev-C 5.11*.

Các bạn có thể dùng chương trình sau chạy và kiểm tra kết quả:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18 | #include <stdio.h>  #include <limits.h>  // limits for interger  #include <float.h>   // limits for float    int main()  {      printf("TYPE                %6s %20s %20s\n", "SIZE", "MIN VALUE", "MAX VALUE");      printf("char:          %6ld byte %20d %20d\n", sizeof(char), CHAR\_MIN, CHAR\_MAX);      printf("unsigned char: %6ld byte %20d %20d\n", sizeof(unsigned char), 0, UCHAR\_MAX);      printf("short:         %6ld byte %20d %20d\n", sizeof(short), SHRT\_MIN, SHRT\_MAX);      printf("int:           %6ld byte %20d %20d\n", sizeof(int), INT\_MIN, INT\_MAX);      printf("long:          %6ld byte %20ld %20ld\n", sizeof(long), LONG\_MIN, LONG\_MAX);      printf("long long:     %6ld byte %20lld %20lld\n", sizeof(long long), LLONG\_MIN, LLONG\_MAX);      printf("float:         %6ld byte %20e %20e\n", sizeof(float), FLT\_MIN, FLT\_MAX);      printf("double:        %6ld byte %20e %20e\n", sizeof(double), DBL\_MIN, DBL\_MAX);      printf("long double:   %6ld byte %20Le %20Le\n", sizeof(long double), LDBL\_MIN, LDBL\_MAX);      return 0;  } |

**5. Tên**

Mỗi chúng ta ai cũng có tên, do vậy trong lập trình C các thành phần cũng sẽ có tên. Ví dụ như tên hàm, tên biến, tên hằng số, tên nhãn,…

***Tên hợp lệ là một chuỗi liên tiếp gồm ký tự chữ, số hoặc dấu gạch dưới. Tên phân biệt chữ hoa chữ thường và không trùng với từ khóa.***

* Tên phân biệt chữ hóa chữ thường nên: **a** khác **A**, **quan** khác **Quan**.
* **VD Các tên đúng:** a, quan, nguyenvanquan7826, quan\_7826, \_7826, \_nhung, \_123huong.
* \*\*VD Các tên sai: \*\*

| **Tên sai** | **Nguyên nhân** |
| --- | --- |
| 1 | Bắt đầu bằng số |
| 1a | Bắt đầu bằng số |
| quan 7826 | Chứa dấu cách |
| quan-7826 | Chứa dấu gạch ngang |
| f(x) | Chứa dấu ngoặc nhọn |
| int | Trùng với từ khóa |

**6. Biến**

Tương ứng với mỗi kiểu dữ liệu chúng ta có các biến thuộc các kiểu đó và có miền giá trị tương ứng như trên dùng để lưu giá trị. Các bạn cần phân biệt kiểu và biến.

VD cái rổ A để đựng rau muống, cái rổ B để đựng rau cần và chúng cùng là rổ, thì tương ứng biến a lưu giá trị số 5, còn biến b lưu giá trị số 9 mặc dù chúng cùng kiểu số nguyên.

Biến có thể thay đổi giá trị trong quá trình chạy chương trình.

**6.1 Khai báo biến**

Để khai báo biến ta dùng cú pháp:

**Kiểu\_dữ\_liệu Danh\_sách\_tên\_biến;**

Lý thuyết thì hơi dài, các bạn xem và chạy chương trình sau sẽ thấy cụ thể hơn:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include <stdio.h>    int main()  {      int a, b; // khai bao 2 bien kieu so nguyen      float c; // khai bao 1 bien kieu so thuc        // Gan gia tri cho cac bien        a = 1;      b = 2;      c = 3.4;        // vua khai bao bien vua gan gia tri      int e = 4, f = 6;        printf("a = %d; b = %d\n", a, b);      printf("c = %f\n", c);      printf("e = %d; f = %d\n", e, f);        return 0;  } |

Như trên các bạn thấy cấu trúc khai báo rồi đó. Còn về lệnh xuất ra màn hình các giá trị thì có một số điều khác đó là cách sử dụng %d, %f . Cái này mình sẽ nói cụ thể ở phần sau, đến đây các bạn cứ viết theo để thấy được cách khai báo biến là ok roài. !

**6.2 Phạm vi của biến**

Khi lập trình, bạn phải nắm rõ phạm vi của biến. Nếu khai báo và sử dụng không đúng, không rõ ràng sẽ dẫn đến sai sót khó kiểm soát được, vì vậy bạn cần phải xác định đúng vị trí, phạm vi sử dụng biến trước khi sử dụng biến.

Mỗi biến được khai báo ở đâu thì sẽ được dùng và có phạm vi hoạt động tương ứng từ nơi nó khai báo đến dấu đóng ngoặc nhọn **}** ngang cấp với nó.

Ví dụ chương trình sau.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | #include <stdio.h>    int a = 1, b = 5; // khai bao bien toan cuc, no se duoc dung o bat ky dau ke tu dong nay.    int main()  {      // khai bao 2 bien trong ham main, no se duoc dung trong toan bo ham main ke tu dong nay      int c = 4, d = 6;        if(c < d)      {          int e = 6, d = 8;  // khai bao bien e va d, no duoc dung trong doan nay.          c = 7;          printf("gia tri cac bien trong khoi:\n");          printf("e = %d \t d = %d \t c = %d\n", e, d, c);      } // den day bien e, d vua khai bao khong con hoat dong nua.        // printf("gia tri bien e = %d\n", e);  // lenh nay sai vi bien e khong con ton tai nua        printf("gia tri cac bien trong ham main:\n");      printf("c = %d \t d = %d\n", c, d);        printf("gia tri cac bien toan cuc:\n");      printf("a = %d \t b = %d\n", a, b);        return 0;  } |

Các bạn chạy chương trình, xem kết quả và tự rút ra nhận xét nhá.

Trong này có một lời giải thích nhỏ với các bạn sau khi chạy chương trình là chúng ta có **khai báo 2 biến d**, và 2 biến **d** này do nằm trong 2 khối lệnh khác nhau (1 cái nằm trong hàm main, 1 cái nằm trong đoạn lệnh if ) nên chúng hoàn toàn khác nhau, không ảnh hưởng gì đến nhau. Nhưng biến c thì chỉ khai báo 1 lần, nên khi dùng c trong đoạn lệnh if thì chính là dùng biến c của hàm main đã khai báo và biến c đã được thay đổi giá trị.

**7. Hằng**

Hằng cũng giống như biến nhưng không thể thay đổi giá trị. Nếu bạn cố tình gán giá trị mới cho hằng thì sẽ bị sai.

Tên hằng thường viết in hoa để dễ nhận biết và phân biệt với biến.

Cú pháp khai báo hằng: **#define TÊN\_HẰNG giá\_trị** (Lưu ý không có dấu chấm phẩy ở cuối).

Các bạn chạy và xem kết quả của ví dụ sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | #include <stdio.h>    #define AGE\_MAX 150     // hang so  #define C 'a'            // hang ky tu  #define NICK\_NAME "nguyenvanquan7826" // hang chuoi    int main()  {      printf("hang AGE\_MAX = %d\n", AGE\_MAX);      printf("hang C = %c\n", C);      printf("hang NICK\_NAME = %s\n", NICK\_NAME);        // AGE\_MAX = 10; // lenh nay sai vi hang khong the thay doi duoc gia tri      return 0; |

**Lập trình C: Bài 3 – Nhập xuất trong C**

**Nội dung**

* [1. Chuỗi định dạng dữ liệu](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#1_Chuoi_dinh_dang_du_lieu)
* [2. Xuất dữ liệu dùng printf](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#2_Xuat_du_lieu_dung_printf)
  + [Ví dụ 1:](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#Vi_du_1)
  + [Ví dụ 2:](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#Vi_du_2)
  + [Ví dụ 3:](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#Vi_du_3)
* [3. Nhập dữ liệu dùng scanf](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#3_Nhap_du_lieu_dung_scanf)
  + [Ví dụ 1:](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#Vi_du_1-2)
  + [Ví dụ 2:](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#Vi_du_2-2)
* [4. Nhập chuỗi trong C](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#4_Nhap_chuoi_trong_C)
  + [4.1 Lỗi khi nhập chuỗi bằng scanf](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#41_Loi_khi_nhap_chuoi_bang_scanf)
  + [4.2 Hiện tượng trôi lệnh](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#42_Hien_tuong_troi_lenh)
* [5. Giải thích một chút về printf, scanf và stdio.h](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#5_Giai_thich_mot_chut_ve_printf_scanf_va_stdioh)
  + [5.1 Chữ f trong printf và scanf](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#51_Chu_f_trong_printf_va_scanf)
  + [5.2 Thư viện stdio.h ?](https://cachhoc.net/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/#52_Thu_vien_stdioh)

Ở các bài trước, chúng ta đã làm quen với một số chương trình mà in lên màn hình một số thông tin. Bài này chúng ta sẽ tìm hiểu kỹ hơn về cách nhập xuất trong C.

**1. Chuỗi định dạng dữ liệu**

Trước khi đến với phần nhập, xuất dữ liệu cho các biến mình sẽ nói về một số định dạng để nhập và xuất. Sau đây là các dấu mô tả định dạng:

* %c : Ký tự đơn
* %s : Chuỗi
* %d : Số nguyên hệ 10 có dấu
* %f : Số chấm động (VD 5.54 khi in sẽ ra 5.540000)
* %e : Số chấm động (ký hiệu có số mũ)
* %g : Số chấm động (VD 5.54 khi in sẽ in ra 5.54)
* %x : Số nguyên hex không dấu (hệ 16)
* %o : Số nguyên bát phân không dấu (hệ 8)
* l : Tiền tố dùng kèm với %d, %x, %o để chỉ số nguyên dài (ví dụ%ld)

**2. Xuất dữ liệu dùng printf**

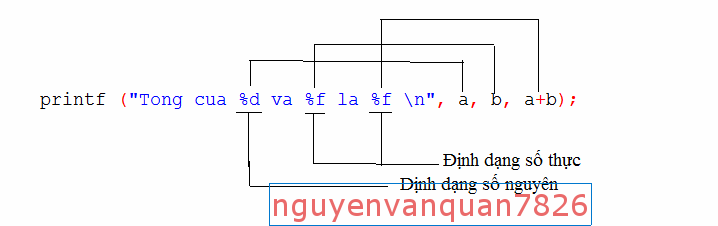
Chúng ta sử dụng hàm **printf** để xuất dữ liệu ra màn hình console (từ print có nghĩa là in). Kết hợp với các chuỗi định dạng ở trên, chúng ta xét một ví dụ đơn giản sau.

**Ví dụ 1:**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | #include <stdio.h>    int main()  {      int a = 12;      float b = 13.5;      char c = 'Q';      long d = 3454;      char\* s = "nguyenvanquan7826"; // khai bao kieu chuoi        printf("Vi du ve su dung lenh printf\n");      printf("Tong cua %d va %f la %f \n", a, b, a+b);      printf("Tich cua %d va %ld la %ld \n", a, d, a\*d);      printf("Ky tu c la: %c \n", c);      printf("Chuoi s la: %s \n", s);      printf("Dinh dang so mu cua b la %e \n", b);      printf("So he 16 va he 8 cua %d la %x va %o \n", a, a, a);      printf("Ma ASCII cua %c la %d", c, c);        return 0;  } |

Ở trên mình có dùng một số ký tự điều khiển (\n – xuống dòng) đã được đề cập ở bài trươc, các bạn có thể xem lại nếu không nhớ. Các bạn chạy chương trình và rút ra nhận xét cho riêng mình.

Mình giải thích 1 câu lệnh để làm rõ hơn việc xuất của chúng ta.



*Câu lệnh printf để in các biến tương ứng*

Chuỗi định dạng được đặt trong ngoặc kép: **“”**. Tương ứng với mỗi định dạng là một biến có kiểu tương ứng, nếu khác kiểu sẽ dẫn đến sai sót.

Các bạn cũng chú ý là đối với số nguyên và ký tự có sự qua lại với nhau thông qua mã ASCII nên chúng ta có thể in mã của ký tự bằng định dạng %d và cũng có thể in ký tự có mã là số nào đó thông qua định dạng %c. Tuy nhiên bản chất của biến không thay đổi. Ở Vd trên câu lệnh in mã ASCII của c sẽ cho số nguyên nhưng bản chất c vẫn là một biến kiểu char.

**Ví dụ 2:**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | #include <stdio.h>    int main()  {      int a = 12;      float b = 13.5;      char c = 'Q';      long d = 3454;      char s[] = "nguyenvanquan7826"; // khai bao kieu chuoi        printf("%6d %5.3f %.3f \n", a, b, a+b);      printf("%-5d %5ld %5ld \n", a, d, a\*d);      printf("%5c \n", c);      printf("%30s \n", s);        return 0;  } |

Các bạn thấy ví dụ này chúng ta dùng các định dạng %d, %f nhưng lại chèn thêm các số ở giữa thành %6d, %5.3f. Ý nghĩa của chúng như sau:

* %5c : Xuất ký tự có bề rộng 5
* %5d : Số nguyên có bề rộng 5
* %20s : Xuất chuỗi có bề rộng 20
* %5.3f : Xuất số thực có bề rộng 5 trong đó có 3 số sau dấu phẩy
* %-5d : Số nguyên có bề rộng 5 nhưng căn lề trái

Các bạn chạy chương trình, xem kết quả để thấy rõ hơn. Trong TH nếu bề rộng của chúng ta có kích thước bé hơn độ dài của số thì sao? Ví dụ chúng ta có số int a = 1234 nhưng chỉ in là %2d, rõ ràng dành 2 phần trống là ít hơn số a có 4 chữ số nhưng kết quả là số vẫn được in đầy đủ. Các bạn có thể thử.

**Ví dụ 3:**

Đặt bài toán là giả sử chúng ta có các biến thể hiện ngày tháng năm sinh bất kỳ (ví dụ ngay = 12, thang = 8, nam = 1992). Hãy in ra theo định dạng là dd/mm/yyyy. Nghĩa là chúng ta phải in ra kết quả 12/08/1992.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | #include <stdio.h>    int main()  {      int ngay = 12;      int thang = 8;      int nam = 1992;        printf("KQ: %02d/%02d/%04d\n", ngay, thang, nam);        return 0;  } |

Ở trên, các bạn chú ý chúng ta dùng **%02d** nghĩa là sao? Nhĩa là số in ra sẽ có 2 chữ số, nếu ban đầu số có ít hơn 2 thì sẽ thêm số 0 đằng trước để đủ 2 chữ số. Ví dụ số 8 sẽ thành 08. Nếu dùng %05d thì sẽ là 00008.

**3. Nhập dữ liệu dùng scanf**

Trong các ví dụ trước, chúng ta chỉ khai báo biến và gán giá trị cho biến nhưng như thế làm chương trình không linh hoạt vì chỉ hoạt động với những giá trị đã định trước, muốn hoạt động với các dữ liệu khác nhau chúng ta lại phải sửa lại code, như vậy thì không phải một chương trình tốt. Một chương trình phần mềm làm ra là phải dành cho nhiều bộ giá trị khác nhau có thể dùng được.

Chính vì vậy dữ liệu sẽ được đưa vào trong quá trình phần mềm được dùng và cụ thể đơn giản là nhập dữ liệu vào từ bàn phím. Chúng ta sử dụng lệnh **scanf** để nhập dữ liệu từ bàn phím (từ scan có nghĩa là quét và chúng ta dùng để quét dữ lệu từ bàn phím).

**Ví dụ 1:**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | #include <stdio.h>    int main()  {      int a;      float b;        printf("Nhap so nguyen a = ");      scanf("%d", &a);        printf("Nhap so thuc b = ");      scanf("%f", &b);        printf("a = %d \t b = %.3f", a, b);        return 0;  } |

Việc sử dụng lệnh **scanf** kết hợp với chuỗi định dạng để nhập dữ liệu sẽ tương tự như khi xuất dữ liệu bằng lệnh **printf**.

**Chú ý đừng quên ký tự & trước mỗi biến khi nhập. Nếu không sẽ sai.**

**Ví dụ 2:**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13 | /\* Tinh dien tich hinh chu nhat \*/  #include <stdio.h>    int main()  {      int a, b;      printf("Nhap chieu dai, chieu rong: \n");      scanf("%d%d", &a, &b);        printf("Dien tich HCN: %d\n", a \* b);        return 0;  } |

Ở ví dụ trên, các bạn thấy chúng ta có thể dùng 1 lệnh scanf để nhập dữ liệu cho nhiều biến, mỗi biến tương ứng với 1 định dạng nhất định.

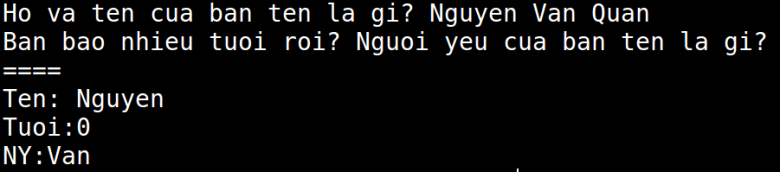
**4. Nhập chuỗi trong C**

**4.1 Lỗi khi nhập chuỗi bằng scanf**

Nếu các bạn dùng hàm scanf để nhập chuỗi thì bạn sẽ thấy rằng không thể nhập được chuỗi có dấu cách hoặc nếu trước đó bạn nhập số thì sau đó không nhập được chuỗi nữa. Nếu không tin bạn có thể thử chạy với chương trình sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include <stdio.h>    int main()  {      int tuoi = 0;      // khai bao chuoi co toi da 30 ky tu      char ten[30], tenNguoiYeu[30];        printf("Ho va ten cua ban ten la gi? ");      scanf("%s", ten); // nhap chuoi khong can dau &        printf("Ban bao nhieu tuoi roi? ");      scanf("%d", &tuoi);        printf("Nguoi yeu cua ban ten la gi? ");      scanf("%s", tenNguoiYeu);        printf("\n====\n");      printf("Ten: %s \nTuoi:%d \nNY:%s \n", ten, tuoi, tenNguoiYeu);        return 0;  } |

Kết quả là bạn sẽ không nhập được tuổi và tên người yêu như hình sau.



*Kết quả khi dùng scanf để nhập chuỗi*

Lý do là scanf chỉ đọc được dữ liệu không có khoảng trắng (đấu cách, dấu tab, enter, …) và các khoảng cách này sẽ được lưu vào bộ đệm bàn phím do đó bạn chỉ nhận được chuỗi đầu tiên trước đấu cách mà thôi (chữ **Nguyen**), sau mỗi dấu cách, các giá trị tiếp theo nếu phù hợp với kiểu dữ liệu của các biến tiếp theo thì nó sẽ gán luôn cho chúng và bạn sẽ không được nhập nữa. Do **tuoi** kiểu nguyên nên không nhận được, **tenNguoiYeu** sẽ nhận giá trị tiếp theo trong các giá trị nhận được là chữ **Van**.

**4.2 Hiện tượng trôi lệnh**

Hiện tượng như trên được gọi là hiện tượng trôi lệnh. Nếu bây giờ bạn thực hiện cho nhập số trước và chuỗi ngay sau đó thì hiện tượng này cũng xảy ra vì **scanf** chỉ đọc số theo đúng định dạng mà không đọc được phím enter khi bạn nhấn lúc nhập xong số (enter là ký tự hoặc cũng có thể coi là chuỗi), nó được lưu vào bộ đệm và khi đọc giá trị nhập cho chuỗi nó tìm trong bộ đệm thấy ký tự enter là kiểu chuỗi nên nó gán luôn cho chuỗi đó.

Để nhập được chuỗi có khoảng trắng (dấu cách) chúng ta sử dụng hàm gets.

Để không bị trôi lệnh khi nhập số trước và chuỗi sau ta cần xóa bộ đệm bàn phím bằng lệnh **fflush(stdin);** ngay sau khi nhập số.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <stdio.h>    int main()  {      int tuoi = 0;      // khai bao chuoi co toi da 30 ky tu      char ten[30], tenNguoiYeu[30];        printf("Ho va ten cua ban ten la gi? ");      gets(ten); // nhap chuoi khong can dau &        printf("Ban bao nhieu tuoi roi? ");      scanf("%d", &tuoi);      fflush(stdin);        printf("Nguoi yeu cua ban ten la gi? ");      gets(tenNguoiYeu);        printf("\n====\n");      printf("Ten: %s \nTuoi:%d \nNY:%s \n", ten, tuoi, tenNguoiYeu);        return 0;  } |

Nếu bạn dùng Linux thì **fflush(stdin);** sẽ không hoạt động, bạn có thể khắc phục bằng cách nhập vào 1 chuỗi tạm ngay sau khi nhập số. Việc nhập chuỗi tạm này chỉ để xóa bỏ các ký tự, chuỗi ký tự thừa có trong bộ đệm như sau (cách này cũng dùng được khi bạn làm trên windows).

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <stdio.h>    int main()  {      int tuoi = 0;      // khai bao chuoi co toi da 30 ky tu      char ten[30], tenNguoiYeu[30], temp[255];        printf("Ho va ten cua ban ten la gi? ");      gets(ten); // nhap chuoi khong can dau &        printf("Ban bao nhieu tuoi roi? ");      scanf("%d", &tuoi);      //fflush(stdin);      gets(temp);        printf("Nguoi yeu cua ban ten la gi? ");      gets(tenNguoiYeu);        printf("\n====\n");      printf("Ten: %s \nTuoi:%d \nNY:%s \n", ten, tuoi, tenNguoiYeu);        return 0;  } |

**5. Giải thích một chút về printf, scanf và stdio.h**

**5.1 Chữ f trong printf và scanf**

Như các bạn đã biết **print** nghĩa là **in**, **scan** là **quét** hay ta gọi trong này là để nhập. Vậy tại sao chúng lại có chữ **f**đằng sau để thành **printf** và **scanf** ?

Chữ **f** này có nghĩa là **format** (định dạng). Như các bạn thấy chúng ta nhập hoặc xuất các giá trị của các biến đều có định dạng % gì đó ví dụ %d là số nguyên, %f là số thực,… và chữ **f** này chính là có ý nghĩa như thế.

**5.2 Thư viện stdio.h ?**

Trong tất cả các chương trình từ trước tới giờ, chúng ta luôn có #include <stdio.h>, vậy nó là gì?

#include nghĩa là bao gồm tức là chương trình của chúng ta sẽ khai báo sử dụng một cái gì đó, mà ở đây là sử dụng thư viện stdio.h

Vậy **stdio.h** là gì? **std** viết tắt của **standard** , **i** viết tắt của **input**, **o** viết tắt của **output**, **h** viết tắt của **header** (header – đầu, trên đầu – vì nó được khai báo ở trên đầu của các chương trình). Vậy nó có nghĩa là **standard input output – nhập xuất chuẩn**. Chúng ta hiểu nó là thư viện phục vụ cho việc nhập xuất chuẩn của chương trình. Nhập xuất chuẩn chính là nhập từ bàn phím và xuất ra màn hình. Có còn rất nhiều nguồn nhập xuất như nhập từ file, nhập từ con trỏ chuột,… xuất ra máy in, xuất ra file,… nhưng họ coi **bàn phím và màn hình là hệ thống nhập xuất chuẩn**.

**Bài tập**

1. Viết chương trình nhập vào một số a bất kỳ và in ra giá trị a2, a3, a4
2. Viết chương trình đọc từ bàn phím 3 số nguyên biểu diễn ngày, tháng, năm và xuất ra màn hình dưới dạng “dd/mm/yyyy”.
3. Viết chương trình đọc và 2 số nguyên và in ra kết quả của phép (+), phép trừ (-), phép nhân (\*), phép chia (/). Nhận xét kết quả chia 2 số nguyên.
4. Viết chương trình nhập vào bán kính hình cầu, tính và in ra diện tích, thể tích của hình cầu đó.
5. Hướng dẫn: S = 4πR^2 và V = (4/3)πR^3.
6. Nhập vào một số là số giây, đổi số giây này ra giờ phút giây và xuất theo dạng gio:phut:giay, mỗi thành phần có 2 chữ số. Ví dụ 3661 = 01:01:01.

**Lập trình C: Bài 4 – Phép toán, toán tử trong C**

Bài học hôm nay sẽ giới thiệu tới các bạn cách thực hiện các phép toán trong C, các toán tử để xử lý dữ liệu nữa.

**Nội dung**

* [1. Toán tử toán học](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#1_Toan_tu_toan_hoc)
* [2. Toán tử quan hệ](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#2_Toan_tu_quan_he)
* [3. Toán tử luận lý](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#3_Toan_tu_luan_ly)
* [4. Toán tử tăng giảm](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#4_Toan_tu_tang_giam)
* [5. Toán tử gán](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#5_Toan_tu_gan)
* [6. Một số toán tử khác](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#6_Mot_so_toan_tu_khac)
  + [6.1 Toán tử phẩy](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#61_Toan_tu_phay)
  + [6.2 Toán tử lấy kích thước](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#62_Toan_tu_lay_kich_thuoc)
* [7. Độ ưu tiên của các toán tử](https://cachhoc.net/2014/12/10/lap-trinh-c-bai-4-phep-toan-toan-tu-trong-c/#7_Do_uu_tien_cua_cac_toan_tu)

**1. Toán tử toán học**

Ngôn ngữ C cung cấp 5 toán tử toán học cơ bản

| **STT** | **Toán tử** | **Tên** | **Ví dụ** | **Kết quả** |
| --- | --- | --- | --- | --- |
| 1 | + | Cộng | 3 + 4 | 7 |
| 2 | – | Trừ | 10.5 – 3 | 7.5 |
| 3 | \* | Nhân | 3 \* 2 | 6 |
| 4 | / | Chia | 6 / 3 | 2 |
| 5 | % | Chia lấy dư | 8 % 3 | 2 |

Ok. Giờ thử làm ví dụ:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | #include <stdio.h>    int main()  {      int a = 5, b = 7;      double c = 4.5, d = 6;        printf("%d + %f = %f \n", a, c, a + c);      printf("%d - %d = %d \n", a, b, a - b);      printf("%d \* %f = %f \n", b, d, b \* d);        /\* Luu y phep chia nhe\*/        printf("%d / %d = %d \n", b, a, b / a);      printf("%f / %d = %f \n", c, a, c / a);      printf("%f / %f = %f \n", c, d, c / d);        printf("%d %% %d = %d \n", b, a, b % a);        return 0;  } |

Kết quả:

5 + 4.500000 = 9.500000

5 – 7 = -2

7 \* 6.000000 = 42.000000

7 / 5 = 1

4.500000 / 5 = 0.900000

4.500000 / 6.000000 = 0.750000

7 % 5 = 2

Toán tử lấy phần dư (%) yêu cầu cả hai toán hạng là số nguyên. Nó trả về phần dư còn lại của phép chia. Ví dụ 7 % 5 được tính toán bằng cách chia số nguyên 7 cho 5 để được 1 và phần dư là 2; vì thế kết quả là 2.

**Thông thường, nếu cả hai toán hạng là số nguyên sau đó kết quả sẽ là một số nguyên**. Tuy nhiên, một hoặc cả hai toán hạng là số thực thì sau đó kết quả sẽ là một số thực.

Khi cả hai toán hạng của toán tử chia là số nguyên thì sau đó phép chia được thực hiện như là một phép chia số nguyên và không phải là phép chia thông thường mà chúng ta sử dụng. Phép chia số nguyên luôn cho kết quả là phần nguyên của thương. Ví dụ: 7 / 5 = 1 chứ không phải 7 / 5 = 1.4. Để khắc phục lỗi này thì ta có thể chuyển một số hoặc cả 2 số sang kiểu thực rồi thực hiện phép chia. Cách chuyển kiểu (hay ép kiểu) ta như sau:

(kiểu cần chuyển) biến.

VD: (float) a;

**Lưu ý** khi ép kiểu thế này thì kiểu của các biến ban đầu không thay đổi mà chỉ là giá trị tức thời (tại thời điểm đó thay đổi sang kiểu mới). Để lưu lại giá trị tức thời này bạn cần khai báo thêm một biến mới có kiểu cần chuyển và gán giá trị đó lại. Ví dụ cho dễ.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include <stdio.h>    int main()  {      int a = 5, b = 7;      double c;        printf("%d / %d = %d \n", b, a, b / a);        /\* Chuyen gia tri tuc thoi cua b sang kieu so thuc\*/      printf("%d / %d = %f \n", b, a, (double)b / a);        /\* Chuyen gia tri tuc thoi cua a sang kieu so thuc\*/      printf("%d / %d = %f \n", b, a, b / (double)a);        /\* Neu lam the nay thi van khong dung, vi b/a duoc so nguyen       \* sau do chung ta moi ep kieu so nguyen do sang so thuc       \*/      printf("%d / %d = %f \n", b, a, (double)(b / a));        return 0;  } |

Kết quả:

7 / 5 = 1

7 / 5 = 1.400000

7 / 5 = 1.400000

7 / 5 = 1.000000

**2. Toán tử quan hệ**

Ngôn ngữ C cung cấp 6 toán tử quan hệ để so sánh các số. Các toán tử quan hệ có giá trị **1 (khi kết quả đúng) hoặc 0 (khi kết quả sai)**.

| **STT** | **Toán tử** | **Tên** | **Ví dụ** | **Kết quả** |
| --- | --- | --- | --- | --- |
| 1 | == | So sánh bằng | 5 == 5 | 1 |
| 2 | != | So sánh khác | 5 != 5 | 0 |
| 3 | > | So sánh lớn hơn | 5 > 4 | 1 |
| 4 | < | So sánh nhỏ hơn | 5 < 4 | 0 |
| 5 | >= | So sánh lơn hơn hoặc bằng | 5 >= 4 | 1 |
| 6 | <= | So sánh nhỏ hơn hoặc bằng | 5 <= 5 | 1 |

Chúng ta sử dụng các toán tử này để so sánh các giá trị, các ký tự,… tuy nhiên **không so sánh các xâu với nhau** vì điều này sẽ dẫn đến các địa chỉ của chuỗi được so sánh chứ không phải là nội dung chuỗi. Chúng ta có các hàm so sánh xâu ở thư viện string và sẽ tìm hiểu sau. Khi so sánh các ký tự với nhau thì bản chất ta có thể hiểu là máy so sánh các mã ASCII của các ký tự với nhau. VD. ‘A’ >’B’ sẽ trả về giá trị 0 vì ‘A’ có mã ASCII là 65 còn ‘B’ là 66.

**3. Toán tử luận lý**

C cung cấp 3 toán tử luận lý cho việc kết nối các biểu thức luận lý. Giống như các toán tử quan hệ, các toán tử luận lý có giá trị là 1 hoặc 0.

| **STT** | **Toán tử** | **Tên** | **Ví dụ** | **Kết quả** |
| --- | --- | --- | --- | --- |
| 1 | ! | Phủ định | ! (5 > 4) | 0 |
| 2 | && | Và | 5 > 4 && 5 > 6 | 0 |
| 3 | || | Hoặc | 5 > 4 || 5 > 6 | 1 |

**4. Toán tử tăng giảm**

Các toán tử tăng một (++) và giảm một (- -) cung cấp các tiện lợi tương ứng cho việc cộng thêm 1 vào một biến số hay trừ đi 1 từ một biến số.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20 | #include <stdio.h>    int main()  {      int i, k;        i = 5; k = i++;      printf("i = %d, k = %d\n", i, k);        i = 5; k = ++i;      printf("i = %d, k = %d\n", i, k);        i = 5; k = i--;      printf("i = %d, k = %d\n", i, k);        i = 5; k = --i;      printf("i = %d, k = %d\n", i, k);        return 0;  } |

Kết quả:

i = 6, k = 5

i = 6, k = 6

i = 4, k = 5

i = 4, k = 4

**Tức là ta có:**

++i và –i thì i được tính trước sau đó sẽ lấy kết quả để thực hiện biểu thức

i++ và i– thì i được đưa vào thực hiện biểu thức trước sau đó mới tính i

**5. Toán tử gán**

Toán tử gán được sử dụng để lưu trữ giá trị cho 1 biến nào đó.

| **Toán tử** | **Ví dụ** | **Tương đương với** |
| --- | --- | --- |
| = | x = 5 | Gán 5 cho x |
| += | x += 5 | x = x + 5 |
| -= | x -= 5 | x = x – 5 |
| \*= | x \*= 5 | x = x \* 5 |
| /= | x /= 5 | x = x / 5 |
| %= | x %= 5 | x = x % 5 |

Chạy test thử nhé.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <stdio.h>    int main()  {      int x;        x = 5;      printf("x = 5 => x = %d\n", x);        x += 5;      printf("x += 5 => x = %d\n", x);        x -= 5;      printf("x -= 5 => x = %d\n", x);        x \*= 5;      printf("x \*= 5 => x = %d\n", x);        x /= 5;      printf("x /= 5 => x = %d\n", x);        return 0;  } |

Kết quả:

x = 5 => x = 5

x += 5 => x = 10

x -= 5 => x = 5

x \*= 5 => x = 25

x /= 5 => x = 5

**6. Một số toán tử khác**

**6.1 Toán tử phẩy**

Nhiều biểu thức có thể được kết nối vào cùng một biểu thức sử dụng toán tử phẩy. Toán tử phẩy yêu cầu 2 toán hạng. Đầu tiên nó ước lượng toán hạng trái sau đó là toán hạng phải, và trả về giá trị của toán hạng phải như là kết quả sau cùng.

Ví dụ:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | #include <stdio.h>    int main()  {      int m, t;        m = (t =2, t\*5 + 10);      printf("t = %d, m = %d\n", t, m);        return 0;  } |

Kết quả:

t = 2, m = 20

**6.2 Toán tử lấy kích thước**

C cung cấp toán tử hữu dụng, sizeof, để tính toán kích thước của bất kỳ hạng mục dữ liệu hay kiểu dữ liệu nào. Nó yêu cầu một toán hạng duy nhất có thể là tên kiểu (ví dụ, int) hay một biểu thức (ví dụ, 100) và trả về kích thước của những thực thể đã chỉ định theo byte. Chạy thử ví dụ nhá. Toán tử này chúng ta đã làm quen ở phần Kiểu dữ liệu rồi.

Chạy ví dụ test thử:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | #include <stdio.h>    int main()  {      printf("char size = %d byte\n", sizeof(char));      printf("short size = %d byte\n", sizeof(short));      printf("int size = %d byte\n", sizeof(int));      printf("long size = %d byte\n", sizeof(long));      printf("float size = %d byte\n", sizeof(float));      printf("double size = %d byte\n", sizeof(double));      printf("1.55 size = %d byte\n", sizeof(1.55));      printf("\"Hello\" size = %d byte\n", sizeof("Hello"));      return 0;  } |

Kết quả:

char size = 1 byte

short size = 2 byte

int size = 4 byte

long size = 8 byte

float size = 4 byte

double size = 8 byte

1.55 size = 8 byte

“Hello” size = 6 byte

**7. Độ ưu tiên của các toán tử**

Độ ưu tiên các toán tử được thực hiện từ trên xuống dưới theo bảng sau. Trong mỗi hàng lại có độ ưu tiên như ở cột thứ 3.

| **Loại** | **Toán tử** | **Thứ tự ưu tiên** |
| --- | --- | --- |
| Postfix | () [] -> . ++ – – | Trái sang phải |
| Unary | + – ! ~ ++ – – (type)\* & sizeof | Phải sang trái |
| Tính nhân | \* / % | Trái sang phải |
| Tính cộng | + – | Trái sang phải |
| Dịch chuyển | << >> | Trái sang phải |
| Quan hệ | < <= > >= | Trái sang phải |
| Cân bằng | == != | Trái sang phải |
| Phép AND bit | & | Trái sang phải |
| Phép XOR bit | ^ | Trái sang phải |
| Phép OR bit | | | Trái sang phải |
| Phép AND logic | && | Trái sang phải |
| Phép OR logic | || | Trái sang phải |
| Điều kiện | ?: | Phải sang trái |
| Gán | = += -= \*= /= %=>>= <<= &= ^= |= | Phải sang trái |
| Dấu phảy | , | Trái sang phải |

**Bài tập**

1. Viết chương trình nhập vào số giây, đổi số giây nhập vào thành dạng “gio:phut:giay”, mỗi thành phần là một số nguyên có 2 chữ số. VD nhập vào 7826 thì in ra 02:10:26
2. Cho biến x = 3; Không chạy chương trình, đoán xem giá trị của x sau khi thực hiện lệnh:
3. **x = ++x – 3 + x;**
4. Sau khi đoán xong thì thử viết chương trình để kiểm tra kết quả.

**Lập trình C: Bài 5 – if else, switch case trong C**

Trong cuộc sống có rất nhiêu điều chúng ta phải lựa chọn, ví như lựa chọn giữa người ta yêu và người yêu ta, chọn nghe theo con tim hay là lý trí,… Nếu hôm nay được nghỉ thì sẽ ở nhà học hay là đi chơi, nếu có người yêu thì ngồi code hay đi mua kem cùng ăn ngoài cổng trường, vân vân và mây mây. Và lập trình là đi từ cuộc sống, phục vụ cuộc sống nên nó cũng phải có những thứ như vậy. Bài này chúng ta sẽ tìm hiểu về các lệnh để phục vụ những lựa chọn như vậy, các lệnh này gọi là **lệnh rẽ nhánh**.

**Nội dung**

* [1. Lệnh và khối lệnh](https://cachhoc.net/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#1_Lenh_va_khoi_lenh)
* [2. Lệnh if và if else](https://cachhoc.net/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#2_Lenh_if_va_if_else)
  + [2.1 Lệnh if](https://cachhoc.net/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#21_Lenh_if)
  + [2.2 Lệnh if – else](https://cachhoc.net/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#22_Lenh_if_8211_else)
  + [2.3 Lệnh if else lồng nhau và if-else-if](https://cachhoc.net/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#23_Lenh_if_else_long_nhau_va_if-else-if)
  + [2.4 Toán tử điều kiện – If else rút gọn](https://cachhoc.net/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#24_Toan_tu_dieu_kien_8211_If_else_rut_gon)
* [2. Lệnh switch case](https://cachhoc.net/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#2_Lenh_switch_case)
* [3. Khi nào dùng if-else, khi nào dùng switch-case](https://cachhoc.net/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#3_Khi_nao_dung_if-else_khi_nao_dung_switch-case)

Trước khi tìm hiểu về cấu trúc các lệnh rẽ nhánh, chúng ta nên tìm hiểu một chút về lệnh đơn và khối lệnh.

**1. Lệnh và khối lệnh**

**Lệnh** là một tác vụ, biểu thức, hàm, cấu trúc điều khiển… nào đó đơn lẻ.

Ví dụ:

|  |  |
| --- | --- |
| 1  2 | x = x + 2; // đây là một lệnh đơn  printf("Day la mot lenh\n");  // đây cũng là một lệnh đơn. |

**Khối lệnh:** là một dãy các câu lệnh được bọc bởi cặp dấu **{ }**.

Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5 | { //dau khoi      a = 78;      b = 26;      printf("Tong %d + %d = %d", a, b, a+b);  } //cuoi khoi |

Khi muốn thực hiện một dãy các câu lệnh liên tiếp nào đó thì chúng ta dùng khối lệnh, tức là đặt chúng vào cặp dấu **{}** và viết thụt vào 1 tab cho dễ nhìn.

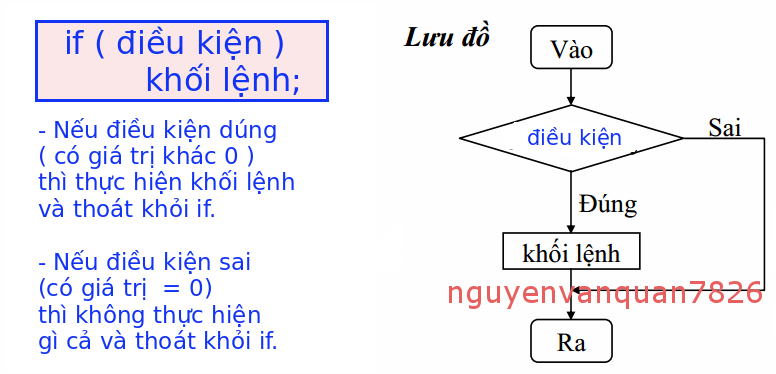
**Lưu ý:** Khi một lệnh được đặt trong cặp ngoặc {} thì cũng được coi là khối lệnh.

**2. Lệnh if và if else**

**2.1 Lệnh if**

Lệnh if dịch ra có nghĩa là nếu cái này thì làm cái kia. Ví dụ nếu bạn có người yêu thì chắc bạn sẽ đi chơi cùng ẻm chứ không ngồi xem bài của mình…

**Cú pháp của lệnh if**



*Cú pháp, lưu đồ và cách hoạt động của lệnh if*

**Chú ý:**

**Không đặt dấu chấm phẩy sau câu lệnh if. Ví dụ: if (a > 0);**

**Khi đặt dấu chấm phẩy ở lệnh if thì coi như kết thúc lệnh if tại đó nên khối lệnh sẽ không được thực hiện dù điều kiện đúng hay sai**.

**Ví dụ:** Ban đầu bạn có 100 (tỷ), hãy nhập vào một số tiền. Nếu số tiền là số dương thì cộng thêm vào ví của bạn ngược lại thì không làm gì cả. In ra số tiền bạn có sau khi nhập.

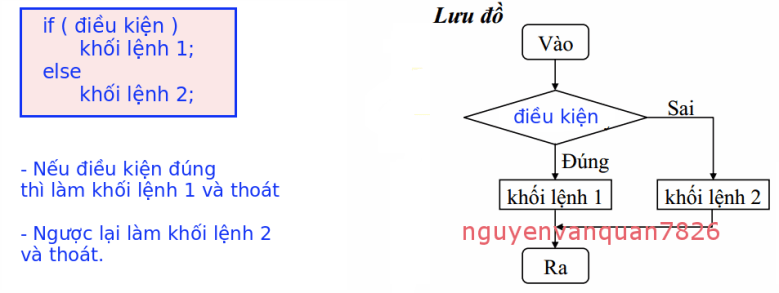
|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19 | #include <stdio.h>    int main()  {      int x = 100; // So tien ban dau ban co      int y; // so tien nhap moi        printf("Nhap so tien = ");      scanf("%d", &y);        if( y > 0 ) // neu so tien nhap vao lon hon 0 thi cong vao vi      {          x = x + y;      }        printf("So tien sau = %d \n", x);        return 0;  } |

Thật đơn giản. Bạn chạy chương trình và xem cách nó hoạt động nhé.

**2.2 Lệnh if – else**

Lệnh if-else là dạng đủ của lệnh if. if là nếu, else là ngược lại.

**Cú pháp lệnh if – else**



*Cú pháp, lưu đồ và cách hoạt động lệnh if – else*

**Ví dụ:** Tương tự ví dụ trên. Ban đầu bạn có 100 (tỷ), vợ bạn có 50 (tỷ), hãy nhập vào một số tiền. Nếu số tiền là số dương thì cộng thêm vào ví của bạn một nửa, của vợ bạn một nửa, ngược lại thì chỉ cộng vào ví của bạn (bản chất là trừ đi vì số tiền có thể âm hoặc bằng 0). In ra số tiền bạn và vợ bạn có sau khi nhập.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <stdio.h>    int main()  {      float x = 100, y = 50; // So tien ban va vo ban co      float z; // so tien nhap moi        printf("Nhap so tien = ");      scanf("%f", &z);        if( z > 0 ) // neu so tien nhap vao lon hon 0 thi cong vao vi      {          x = x + z / 2;          y = y + z / 2;      } else      {          x = x + z;      }        printf("So tien cua ban = %.2f \n", x);      printf("So tien cua vo ban = %.2f \n", y);        return 0;  } |

Khá là đơn giản. Bạn thử chạy và xem kết quả nhé. (Ở bài này mình dùng kiểu float vì có phép chia nên số tiền có thể lẻ).

**2.3 Lệnh if else lồng nhau và if-else-if**

Trong nhiều trường hợp, chúng ta cần phải xử lý nhiều điều kiện chứ không chỉ có 1 điều kiện. Do vậy chúng ta có thể lồng nhiều câu lệnh if else vào với nhau.

**Ví dụ:** Công ty đang triển khai một loại cân sao cho cân theo cặp phục vụ cho cân các cặp tình nhân, và thông báo ra bạn trai nặng hơn hay bạn gái nặng hơn hay hai bạn nặng bằng nhau.

Như vậy chúng ta cần xử lý 3 trường hợp chứ không phải chỉ đơn giản là nếu và ngược lại nữa. Chúng ta có code như sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | #include <stdio.h>    int main()  {      float a; // can nang cua ban trai      float b; // can nang cua ban gai        printf("Nhap vao can nang ban trai va ban gai: \n");      scanf("%f%f", &a, &b);        if( a > b )      {          printf("Ban trai nang hon!\n");      } else      {          if( a < b)          {              printf("Ban gai nang hon\n");          } else          {              printf("Hai ban nang bang nhau\n");          }      }        return 0;  } |

Ok. Vậy là chúng ta sử dụng được chúng lồng vào nhau. Tuy nhiên cách này sẽ làm code hơi rối và tốn giấy mực. Bạn có thể sử dụng lệnh if-else-if như sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <stdio.h>    int main()  {      float a; // can nang cua ban trai      float b; // can nang cua ban gai        printf("Nhap vao can nang ban trai va ban gai: \n");      scanf("%f%f", &a, &b);        if( a > b )      {          printf("Ban trai nang hon!\n");      } else if( a < b)      {          printf("Ban gai nang hon\n");      } else      {          printf("Hai ban nang bang nhau\n");      }        return 0;  } |

Cách code và ý nghĩa cũng rất rõ ràng, dễ hiểu thôi :).

**2.4 Toán tử điều kiện – If else rút gọn**

Chúng ta có một toán tử rút gọn của lệnh if else như sau:

điều kiện ? biểu thứ 1: biểu thức 2;

Nếu điều kiện đúng thì biểu thức 1 được thực hiện và giá trị của biểu thức 1 là giá trị của toàn bộ lệnh. Nếu điều kiện sai thì biểu thức 2 được thực hiện và giá trị của biểu thức 2 trở thành giá trị của toàn bộ lệnh.

Ví dụ:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18 | #include <stdio.h>    int main()  {      int x = 3;      int y = 4;      int z;        z = x > y ? x : y;        printf("z = %d\n", z);        x = x > y ? x : 100;        printf("x = %d\n", x);        return 0;  } |

Kết quả sau khi chạy chương trình:

z = 4

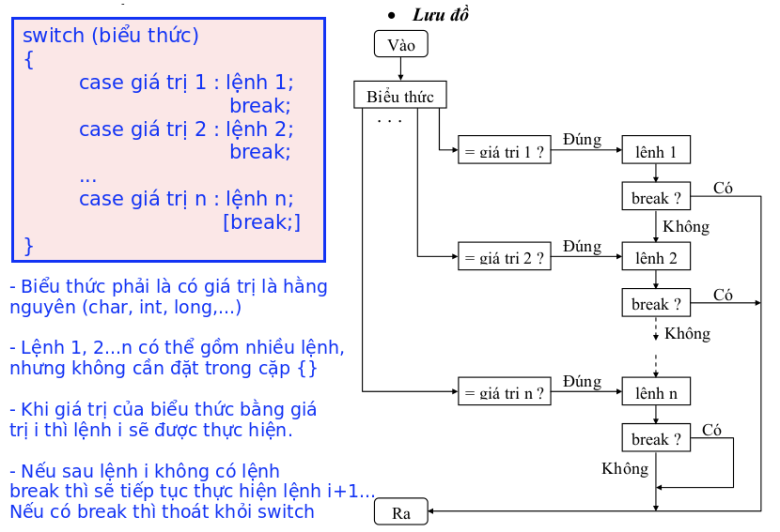
x = 100

Như vậy ta thấy biểu thức 2 và biểu thức 3 có thể là 1 giá trị biến, hằng, hoặc một hàm nào đó có trả về giá trị.

**3. Lệnh switch case**

Lệnh switch cũng giống cấu trúc if, nhưng nó mềm dẻo hơn và linh động hơn nhiều so với sử dụng if. Tuy nhiên, nó cũng có mặt hạn chế là kết quả của biểu thức phải là giá trị hằng nguyên (có giá trị cụ thể). Một bài toán sử dụng lệnh switch thì cũng có thể sử dụng if, nhưng ngược lại thì còn tùy thuộc vào giải thuật của bài toán.

**Cấu trúc lệnh switch case**



*Cú pháp, lưu đồ và cách hoạt động lệnh switch case*

**Ví dụ:** Nhập vào 1 số từ 1->5 và in ra cách đọc tương ứng:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <stdio.h>    int main()  {      int a;      printf("Nhap a = ");      scanf("%d",&a);        switch(a)      {          case 1: printf("Mot\n");                  break;          case 2: printf("Hai\n");                  break;          case 3: printf("Ba\n");                  break;          case 4: printf("Bon\n");                  break;          case 5: printf("Nam\n");                  break;      }        return 0;  } |

Các bạn chạy thử, và thử bỏ một vài lệnh break xem nó ra sao nhé.

Ngoài ra, chúng ta có cấu trúc switch-case-default nữa. Nguyên lý hoạt động giống như switch-case nhưng nếu các trường hợp đều không thỏa mãn thì sẽ thực hiện lệnh của trường hợp default (dịch là trường hợp mặc định).

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | #include <stdio.h>    int main()  {      int a;      printf("Nhap a = ");      scanf("%d",&a);        switch(a)      {          case 1: printf("Mot\n");                  break;          case 2: printf("Hai\n");                  break;          case 3: printf("Ba\n");                  break;          case 4: printf("Bon\n");                  break;          case 5: printf("Nam\n");                  break;          default: printf("Ban da nhap mot so khac\n");                  break;      }        return 0;  } |

**Khi nào dùng if-else, khi nào dùng switch-case ?**

Như mình đã nói ở trên, lệnh switch case có tính mềm dẻo và linh động hơn if else, cách viết cũng thoáng và khó nhầm hơn if else nhưng một bài dùng được switch case thí chắc chắn dùng được bằng if else còn khi dùng if else lại chưa chắc dùng được bằng switch case.

**Vậy các bạn nên dùng switch case khi:**

* Số lượng điều kiện là lớn như dạng menu chọn,… -> switch case viết sẽ lạch mạc, dễ kiểm soát.
* Các trường hợp điều kiện phải có giá trị là số nguyên hoặc ký tự.

**Các trường hợp còn lại, các bạn hãy dùng if – else**.

**Bài tập:**

1. Viết chương trình nhập vào 4 số nguyên. Tìm và in ra số lớn nhất.
2. Hướng dẫn: Ta có 4 số nguyên a, b, c, d. Tìm 2 số nguyên lớn nhất x, y của 2 cặp (a, b) và (c, d). Sau đó so sánh x và y để tìm ra số nguyên lớn nhất.
3. Hãy nhập vào một số là 1 tháng trong năm, in ra xem tháng đó có bao nhiêu ngày. (Coi như không có năm nhuận).
4. Hãy lập trình chương trình giải phương trình bậc 2: ax^{2} + bx + c = 0 với a, b, c nhập từ bàn phím.
5. Viết chương trình nhập vào 3 giá trị nguyên dương a, b, c. Kiểm tra xem a, b, c có phải là 3 cạnh của tam giác không? Nếu là 3 cạnh của tam giác thì tính diện tích của tam giác theo công thức:

S = \sqrt{p(p-a)(p-b)(p-c)} với p là nửa chu vi tam giác. p = \frac{a+b+c}{2}

1. Viết chương trình tính tiền điện gồm các điều kiện sau:

* Tiền thuê bao điện kế: 1000đ/tháng
* Định mức sử dụng điện cho mỗi hộ là: 50 KW với giá 230đ/KW
* Nếu phần vượt định mức <= 50KW thì tính giá 480đ/KW
* Nếu 50KW < phần vượt định mức < 100KW thì tính giá 700đ/K
* Nếu phần vượt định mức >= 100KW thì tính giá 900đ/KW

Chỉ số mới và cũ được nhập vào từ bàn phím. In ra màn hình chỉ số cũ, chỉ số mới, tiền trả định mức, tiền trả vượt định mức, tổng tiền phải trả.

**Lập trình C : Bài 6 – Vòng lặp for trong C**

**Nội dung**

* [1. Ví dụ mở đầu](https://cachhoc.net/2018/01/18/lap-trinh-c-bai-6-vong-lap-trong-c/#1_Vi_du_mo_dau)
* [2. Cấu trúc vòng for](https://cachhoc.net/2018/01/18/lap-trinh-c-bai-6-vong-lap-trong-c/#2_Cau_truc_vong_for)
* [Ví dụ 1:](https://cachhoc.net/2018/01/18/lap-trinh-c-bai-6-vong-lap-trong-c/#Vi_du_1)
* [Ví dụ 2:](https://cachhoc.net/2018/01/18/lap-trinh-c-bai-6-vong-lap-trong-c/#Vi_du_2)
* [Ví dụ 3:](https://cachhoc.net/2018/01/18/lap-trinh-c-bai-6-vong-lap-trong-c/#Vi_du_3)
* [Ví dụ 4:](https://cachhoc.net/2018/01/18/lap-trinh-c-bai-6-vong-lap-trong-c/#Vi_du_4)
* [Ví dụ 5](https://cachhoc.net/2018/01/18/lap-trinh-c-bai-6-vong-lap-trong-c/#Vi_du_5)

**1. Ví dụ mở đầu**

Việc in ra tên của bạn rất đơn giản phải không. Một chương trình viết rất đơn giản.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include <stdio.h>    int main()  {      printf("Nguyen Van Quan\n");      return 0;  } |

Ok. Vậy bây giờ tôi muốn bạn in ra 10 lần tên bạn?

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | #include <stdio.h>    int main()  {      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      printf("Nguyen Van Quan\n");      return 0;  } |

Khá đơn giản, chỉ cần copy thành 10 dòng printf là xong. Nhưng nếu không phải 10, mà là 1000 hoặc nhiều hơn? Rõ ràng việc copy là không khả thi.

Trong thực tế, các phần mềm sẽ có những công việc phải lặp đi lặp lại nhiều lần giống như ví dụ trên. Hoặc ví dụ như phần mềm của bạn cần in một danh sách các nhân viên, mỗi nhân viên có thông tin (nội dung) khác nhau nhưng cấu trúc thì rất giống nhau. Hoặc đơn giản hơn là bạn lướt facebook và thấy các tin bài từ bạn bè của mình, các tin này không giống nhau về nội dung nhưng cấu trúc thì rất giống nhau.

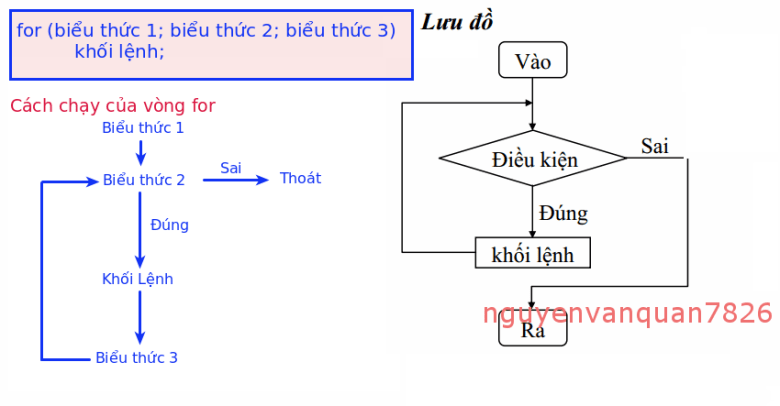
Để thực hiện được những công việc cần lặp lại nhiều lần như thế, chúng ta sẽ sử dụng cấu trúc vòng lặp gồm **for, while, do-while**. Bài này chúng ta sẽ tìm hiểu về vòng lặp **for**.

Chúng ta sẽ sửa lại ví dụ in 10 lần tên bạn như sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | #include <stdio.h>    int main()  {      int i;      for (i = 1; i <= 10; i++)      {          printf("Nguyen Van Quan\n");      }      return 0;  } |

Như code trên, tôi đã sử dụng **vòng lặp for** để thực hiện công việc in ra tên của mình 10 lần, với 1000 lần cũng thật đơn giản khi tôi thay số 10 thành số 1000 (bạn có thể thử). Nhìn vào code bạn cũng có thể hình dung cơ bản về việc chúng ta làm.

**2. Cấu trúc vòng for**



*Cú pháp, lưu đồ và cách hoạt động của vòng lặp for*

Nhìn vào hình trên ta thấy vòng for có một cú pháp và cách hoạt động khá rõ ràng và dễ hiểu. Tuy nhiên các bạn cần lưu ý:

* Sau lệnh for không có chấm phẩy.
* 3 biểu thức của vòng lặp for ngăn cách nhau bằng dấu chấm phẩy (**;**), các biểu thức có thể vắng mặt 1,2 hoặc cả 3 nhưng vẫn phải có dấy chấm phẩy này.
* Biểu thức 1 bao giờ cũng chỉ được tính toán một lần khi gọi thực hiện for.
* Biểu thức 2, 3 và thân for có thể thực hiện lặp lại nhiều lần.
* Biểu thức 1, 2, 3 phải phân cách bằng dấu chấm phẩy ( ; ), các biểu thức có thể vắng mặt 1,2 hoặc cả 3 nhưng vẫn phải có dấy chấm phẩy này.
* Nếu biểu thức 2 không có, vòng for được xem là luôn luôn đúng. Muốn thoát khỏi vòng lặp for phải dùng một trong lệnh break, goto hoặc return.
* Với mỗi biểu thức có thể viết thành một dãy biểu thức con phân cách nhau bởi dấu phẩy. Khi đó các biểu thức con được xác định từ trái sang phải. Tính đúng sai của dãy biểu thức con trong biểu thức thứ 2 được xác định bởi biểu thức con cuối cùng.
* Trong thân for (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác, vòng lặp khác.
* Khi gặp lệnh break, cấu trúc lặp sâu nhất sẽ thoát ra.
* Trong thân for có thểdùng lệnh goto để thoát khỏi vòng lặp đến vị trí mong muốn.
* Trong thân for có thể sử dụng return để trở về một hàm nào đó.
* Trong thân for có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

Quay trở lại code của ví dụ trên:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | #include <stdio.h>    int main()  {      int i;      for (i = 1; i <= 10; i++)      {          printf("Nguyen Van Quan\n");      }      return 0;  } |

Quá trình hoạt động như sau:

* Bước 1: Chạy biểu thức 1 tức là gán i = 1.
* Bước 2: Chạy biểu thức 2 tức là kiểm tra i <= 10 không. Do i = 1 nên cái này đúng => Chạy khối lệnh tức là in ra tên.
* Bước 3: Chạy biểu thức 3 tức là tăng i lên 1 đơn vị. Giờ thì i = 2.
* Bước 4: Quay lại biểu thức 2. Kiểm tra i < 10 không. Do i = 2 nên vẫn đúng => Chạy khối lệnh, tức là in ra tên.
* Bước 5: Chạy biểu thức 3 tức tăng i lên 1. Giờ i = 3.
* Bước 6. Quay lại biểu thức 2,…
* Bước….
* Bước x: Chạy biểu thức 3, tăng i lên 1. Giờ i = 10.
* Bước x+1: Quay lại biểu thức 2 tức kiểm tra i <= 10. Vẫn đúng => Chạy khối lệnh in ra tên.
* Bước x+2: Chạy biểu thức 3, tăng i lên 1. Giờ i = 11.
* Bước x+3: Quay lại biểu thức 2, kiểm tra i <=10. Giờ thì sai vì i đang là 11 => Thoát khỏi vòng for. Thực hiện lệnh tiếp theo ngang hàng vòng for (lệnh return 0 bên dưới).

Ok. Giờ các bạn đã cơ bản hiểu về vòng lặp for. Tiếp tục làm vài ví dụ nữa.

**Ví dụ 1:**

Viết chương trình in ra 15 lần tên của bạn, kèm theo thứ tự như sau:

1. Nguyen Van Quan
2. Nguyen Van Quan
3. Nguyen Van Quan
4. ….

Hum… làm sao lấy được các số tăng dần? Các bạn có nhìn vào quá trình chạy các bước của vòng for? Chúng ta có 1 biến i cứ tăng dần… Vậy chúng ta sẽ lợi dụng nó.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | #include <stdio.h>    int main()  {      int i;      for (i = 1; i <= 10; i++)      {          printf("%d.Nguyen Van Quan\n", i);      }      return 0;  } |

**Ví dụ 2:**

Hãy in ra các số dương chẵn từ 1 đến 20 trên cùng 1 dòng, cách nhau bởi dấu cách.

Chỉ cần cho i chạy từ 1 đến 20 và chúng ta sẽ kiểm tra số nào chẵn thì in ra. Để kiểm tra số chẵn, chúng ta nhớ lại phép chia lấy dư. Nếu i chia 2 dư 0 thì là số chẵn.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | #include <stdio.h>    int main()  {      int i;      for (i = 1; i <= 20; i++)      {          if( i % 2 == 0)          {              printf("%d ", i);          }      }      return 0;  } |

Ok. Tuy nhiên tôi muốn bạn suy nghĩ thêm chút trước khi nhìn xuống gợi ý và code. Làm sao không dùng if kiểm tra số chẵn mà vẫn làm được bài này?

Hãy nhớ các biểu thức trong for không phải là cố định, chúng ta có thể thay đổi nó. Và hãy thay đổi cách tăng i. Chúng ta cho i bắt đầu từ 2 và mỗi lần tăng thì tăng lên 2 đơn vị.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | #include <stdio.h>    int main()  {      int i;      for (i = 2; i <= 20; i = i + 2)      {          printf("%d ", i);      }      return 0;  } |

Rõ ràng code này ngon hơn code của lần trước đó. Ở trên chúng ta tăng i lên 2 đơn vị bằng lệnh **i = i + 2**.

Đây là phép gán i + 2 cho i, chứ không phải lệnh so sánh i và i + 2.

**Ví dụ 3:**

Tính tổng các số nguyên chẵn từ 1 đến 20.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | #include <stdio.h>    int main()  {      int i;      int s = 0;      for (i = 2; i <= 20; i = i + 2)      {          s = s + i;      }        printf("S = %d\n", s);        return 0;  } |

Chú ý trong code trên, chúng ta khai báo 1 biến s (viết tắt của sum) để lưu lại tổng. Mỗi lần i tăng lên thì chúng ta cộng dồn vào s bằng lệnh **s = s + i**.

**Ví dụ 4:**

Tính tổng các số lẻ từ 1 đến 20 nhưng đến khi nào tổng vượt quá 15 thì dừng lại.

Ở đây chúng ta sẽ cộng tổng các số lẻ, giống như các số chẵn thôi, tuy nhiên khi mà tổng lớn hơn 15 rồi thì chúng ta sẽ dừng lại, không cộng nữa.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18 | #include <stdio.h>    int main()  {      int i;      int s = 0;      for (i = 1; i <= 20; i = i + 2)      {          if(s <= 15)          {              s = s + i;          }      }        printf("S = %d\n", s);        return 0;  } |

Đơn giản chỉ việc kiểm tra khi nào s vẫn nhỏ hơn hoặc bằng 15 thì ta cộng vào. Chúng ta sẽ có s = 1 + 3 + 5 + 7 = 16.

Tuy nhiên nếu bạn để ý, khi s > 15, chúng ta không cộng vào s nhưng vòng lặp for vẫn chạy cho đến khi i > 20. Tức là sẽ chạy tiếp các lần lặp i = 9, 11, 13, 15, 17, 19. Do vậy nó làm lãng phí, thừa thãi và làm chương trình chạy lâu hơn. Vì vậy chúng ta nên ngắt vòng lặp tại thời điểm thực hiện xong lệnh khi i = 7. Để thực hiện việc này có thể dùng **break**, hoặc **goto**, nhưng khuyên các bạn nên dùng break vì đơn giản và an toàn.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | #include <stdio.h>    int main()  {      int i;      int s = 0;      for (i = 1; i <= 20; i = i + 2)      {          if(s <= 15)          {              s = s + i;          } else          {              break;          }      }        printf("S = %d\n", s);        return 0;  } |

**Ví dụ 5**

Nhập vào n số nguyên, tính tổng các số chẵn đã nhập.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <stdio.h>    int main()  {      int i, n, x;      int s = 0;        printf("Enter n = ");      scanf("%d", &n);        for (i = 1; i <= n; i++)      {          printf("Enter number %d : ", i);          scanf("%d", &x);            if(x % 2 != 0) continue;            s = s + x;      }        printf("S = %d\n", s);        return 0;  } |

Ở ví dụ này, tôi hướng dẫn các bạn cách dùng lệnh **continue**. Lệnh này không thoát khỏi vòn lặp nhưng lại có tác dụng là quay trở về biểu thức 3 mà không làm các lệnh đằng sau nó. Ở đây nếu x nhập vào không chẵn thì bỏ qua lệnh cộng dồn s mà chạy đến lệnh tăng i ngay.

**Bài tập**:

1. Viết chương trình nhập vào số n và in ra các ước của số n đó.
2. Viết chương trình kiểm tra 1 số có là số nguyên tố không? Số nguyên tố là số nguyên dương có duy nhất 2 ước là 1 và chính nó. Ví dụ số 2, 3, 5, …
3. Viết chương trình kiểm tra 1 số có là số hoàn hảo không? Số hoàn hảo là số nguyên dương có tổng các ước bằng 2 lần nó. VD số 6 có các ước 1, 2, 3, 6 và tổng 1 + 2 + 3 + 6 = 12 (bằng 2 lần 6).
4. Viết chương trình tính S = 1 + 1/2 + 1/3 + … + 1/N
5. Viết chương trình tính tổng bình phương các số lẻ từ 1 đến n.
6. Viết chương trình tính n! biết n! = 1.2.3.4…n
7. Viết chương trình tính số thứ n của dãy fibonaci biết dãy f(n) = f(n-1) + f(n-2), n > 2 và f(1) = 1, f(2) = 1.
8. Viết chương trình nhập vào chiều dài, chiều rộng của hình chữ nhật. Vẽ hình chữ nhật dấu sao (\*) có kích thước đã nhập.

## Lập trình C: Bài 7 – Vòng lặp while, do…while trong C

**Nội dung**

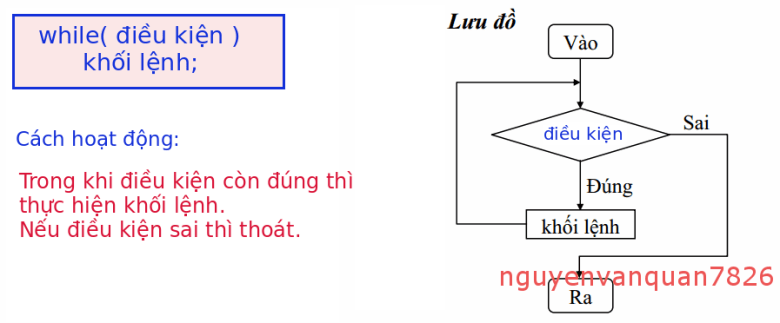
* [1. Vòng lặp while](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/#1_Vong_lap_while)
  + [Cấu trúc vòng lặp while](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/#Cau_truc_vong_lap_while)
  + [Ví dụ 1:](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/#Vi_du_1)
  + [Ví dụ 2:](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/#Vi_du_2)
* [2. Vòng lặp do…while](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/#2_Vong_lap_do8230while)
  + [Cấu trúc vòng lặp do…while](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/#Cau_truc_vong_lap_do8230while)
  + [Ví dụ 1](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/#Vi_du_1-2)
* [3. So sánh, cách dùng vòng lặp for, while, do…while](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/#3_So_sanh_cach_dung_vong_lap_for_while_do8230while)

Ở bài trước, chúng ta đã làm quen với vòng lặp for với những tiện ích cho chúng ta việc lặp đi lặp lại nhiều lần. Bài này chúng ta tiếp tục tìm hiểu 2 loại vòng lặp khác có chức năng tương tự vòng lặp for nhưng cách dùng có khác một chút là vòng lặp **while** và **do…while**.

Các bạn có để ý đến các trang web mà chúng ta cần đăng nhập? Đơn giản như facebook hoặc google. Nếu các bạn viết sai username hoặc password thì sẽ không thể đăng nhập được dù cho bạn nhập bao nhiêu lần đi nữa. Cứ sai là họ bắt chúng ta nhập lại. Rõ ràng ở vòng for chúng ta thường nhìn thấy vòng for lặp một số lần cụ thể như từ 1 đến n,… Nhưng việc đăng nhập thì không biết bao nhiêu lần mới được mà chỉ có là đúng thì mới kết thúc. Đó là cách hoạt động của vòng lặp while và do…while.

## 1. Vòng lặp while

### Cấu trúc vòng lặp while



Cấu trúc, lưu đồ và cách hoạt động vòng lặp while

### Ví dụ 1:

Viết chương trình nhập vào số bất kỳ đến khi nhập số âm thì dừng lại.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | #include <stdio.h>    int main()  {      int x = 1; // make x >= 0 is true -> while will run        while (x >= 0)      {          printf("Nhap vao mot so : ");          scanf("%d", &x);      }        printf("Ok, Program is finish!\n");        return 0;  } |

**Lưu ý:** Ở trên mình vừa khai báo vừa gán giá trị cho biến x là 1 để đảm bảo rằng ban đầu vòng lặp while có điều kiện đúng để thực hiện được các lệnh bên trong, sau đó mỗi lần nhập thì nhập gì x nhận cái đó. Nếu không khởi gán bằng 1 thì có thể x sẽ nhận 1 giá trị ngẫu nhiên và có thể âm làm cho vòng lặp không thực hiện được.

**Kết quả:**

Nhap vao mot so : 4

Nhap vao mot so : 5

Nhap vao mot so : 2

Nhap vao mot so : 0

Nhap vao mot so : 1

Nhap vao mot so : -3

Ok, Program is finish!

### Ví dụ 2:

In các số từ 9 về 0 theo chiều giảm dần. Các số nằm trên 1 dòng.

Thông thường các bạn sẽ nghĩ đến vòng lặp for. Ok chuẩn. Nhưng ở đây mình muốn các bạn biết cách dùng while cũng làm được.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | #include <stdio.h>    int main()  {      int n = 10;        while (n--)      {          printf("%d ", n);      }        printf("\n");        return 0;  } |

Thật đơn giản. Tuy nhiên cần chú ý một chút là điều kiện ở vòng lặp while trong code trên là gì?

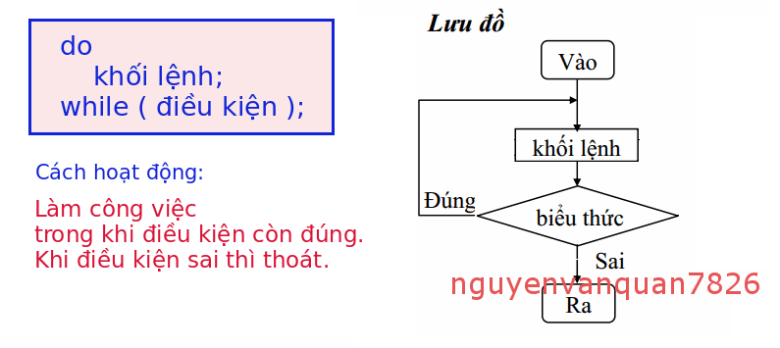
Đó là n có khác 0 hay không. khi viết while (n--) tức là kiểm tra xem n có khác 0 hay không hoặc có thể hiểu trong khi n đúng (đúng là khác 0) thì làm công việc in ra số n. Còn việc n-- thì chỉ là trừ đi n 1 đơn vị thôi. Và có một lưu ý quan trọng trong kết quả:

9 8 7 6 5 4 3 2 1 0

Tại sao số 0 vẫn được in ra trong khi theo phân tích thì khi n = 0 tức điều kiện sai, sai thì phải thoát chứ? Đó là do biểu thức n-- của chúng ta. Khi n = 1 thì n-- sẽ làm n có giá trị 0. Nhưng do biểu thức trừ ở phía sau n nên nó thực hiện sau lệnh kiểm tra n = 0. Tức là khi n = 1 thì biểu thức while(n--) vẫn đúng và sau đó n mới bị giảm về 0 và thực hiện lệnh in, do vậy khi in ra sẽ in cả số 0.

## 2. Vòng lặp do…while

### Cấu trúc vòng lặp do…while



Cấu trúc, lưu đồ và cách hoạt động vòng lặp do…while

### Ví dụ 1

Lưu trước mật khẩu facebook của bạn. Hãy viết chương trình nhập vào mật khẩu để đăng nhập.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18 | #include <stdio.h>  #include <string.h> // for strcmp function    int main()  {      char p[20] = "Iloveyou";      // pass saved      char pass[20];              // pass must enter        do      {          printf("Enter your password: ");          gets(pass);      } while ( strcmp(p, pass) != 0);        printf("Ok. You are login success!\n");        return 0;  } |

Kết quả:

Enter your password: quan

Enter your password: nguyenvanquan

Enter your password: iloveyou

Enter your password: Iloveyou

Ok. You are login success!

**Lưu ý:** Ở code trên, mình có dùng hàm **strcmp để so sánh 2 chuỗi**, hàm này nằm trong thư viện **string.h**

Hàm strcmp so sánh chuỗi p và chuỗi pass và cho ta kết quả:

* 1 Nếu p lớn hơn pass
* 0 Nếu p giống pass
* -1 Nếu p nhỏ hơn pass

Do vậy chúng ta có điều kiện strcmp(p, pass) != 0 tức là 2 chuỗi này không giống nhau.

Chi tiết cách so sánh 2 chuỗi, các bạn tham khảo tại bài [So sánh 2 chuỗi trong C – strcmp](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/2018/01/19/sanh-chuoi-trong-c-strcmp/)

## 3. So sánh, cách dùng vòng lặp for, while, do…while

* Vòng lặp for thường sử dụng khi biết được số lần lặp xác định.
* Vòng lặp thường while, do…while sử dụng khi không biết rõ số lần lặp
* Khi gọi vòng lặp while, do…while, nếu điều kiện sai, vòng lặp while sẽ không được thực hiện lần nào nhưng vòng lặp do…while thực hiện được 1 lần.

**Bài tập**

1. Viết chương trình lặp lại nhiều lần công việc nhập một ký tự và in ra mã ASCII của ký tự đó, khi nào nhập số 0 thì dừng. ( Tham khảo [bảng mã ACSII](https://cachhoc.net/2018/01/19/lap-trinh-c-bai-7-vong-lap-trong-c/2013/08/22/bang-ma-ascii-ascii-table/) )
2. Viết chương trình in ra chữ: “Em có yêu anh không?”, Và cho người dùng nhập c (có) hoặc k (không). Khi nào nhập c thì dừng chương trình, còn nếu nhập k thì cứ bắt nhập lại ^^. (Gửi cho bạn gái/trai)
3. Nhập vào 1 số, phân tích số đó thành tích các thừa số nguyên tố.
4. Viết chương trình tìm USCLN, BSCNN của 2 số.

## Lập trình C: Bài 8 – Hàm trong C

**Nội dung**

* [1. Mở đầu về hàm](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#1_Mo_dau_ve_ham)
* [2. Nguyên tắc hoạt động của hàm](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#2_Nguyen_tac_hoat_dong_cua_ham)
* [3. Cấu trúc của hàm](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#3_Cau_truc_cua_ham)
* [4. Các ví dụ về hàm](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#4_Cac_vi_du_ve_ham)
  + [Ví dụ 1: Hàm không trả về giá trị](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#Vi_du_1_Ham_khong_tra_ve_gia_tri)
  + [Ví dụ 2: Hàm có giá trị trả về](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#Vi_du_2_Ham_co_gia_tri_tra_ve)
  + [Ví dụ 3: Hàm kiểm tra](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#Vi_du_3_Ham_kiem_tra)
  + [Ví dụ 4: Các hàm gọi nhau](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#Vi_du_4_Cac_ham_goi_nhau)
  + [Ví dụ 5: Khai báo hàm trước khi định nghĩa hàm](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#Vi_du_5_Khai_bao_ham_truoc_khi_dinh_nghia_ham)
  + [Ví dụ 6: Hàm đệ quy](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#Vi_du_6_Ham_de_quy)
  + [Ví dụ 7: Dùng dẫn hướng #define để định nghĩa hàm đơn giản](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/#Vi_du_7_Dung_dan_huong_define_de_dinh_nghia_ham_don_gian)

## 1. Mở đầu về hàm

Hãy nhập và chạy 2 ví dụ sau, quan sát kết quả và nhận xét.

**Ví dụ 1:**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | // e.g 1 about function in C - code by nguyenvanquan7826    #include <stdio.h>    void loiKhuyen()  {      printf("Neu hoc nghiem tuc va cham chi thi ban se thay:\n");      printf("Khong co viec gi kho\n");      printf("Chi so long khong ben\n");      printf("Dao nui va lap bien\n");      printf("Quyet chi at lam nen\n");      printf("\n");  }    int main()  {      printf("Hoc bai nay kho qua!\n");      loiKhuyen();        printf("Hoc C kho qua!\n");      loiKhuyen();        return 0;  } |

Kết quả:

Hoc bai nay kho qua!

Neu hoc nghiem tuc va cham chi thi ban se thay:

Khong co viec gi kho

Chi so long khong ben

Dao nui va lap bien

Quyet chi at lam nen

Hoc C kho qua!

Neu hoc nghiem tuc va cham chi thi ban se thay:

Khong co viec gi kho

Chi so long khong ben

Dao nui va lap bien

Quyet chi at lam nen

**Ví dụ 2:**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | // e.g 2 about function in C - code by nguyenvanquan7826    #include <stdio.h>    void vechu(char chu, int soluong)  {      int i;      for (i = 0; i < soluong; i++) {          printf("%c", chu);      }      printf("\n");  }    int main()  {      vechu('h', 20);      vechu('a', 30);      vechu('h', 12);      vechu('a', 5);      vechu('p', 10);        return 0;  } |

Kết quả:

hhhhhhhhhhhhhhhhhhhh

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

hhhhhhhhhhhh

aaaaa

pppppppppp

Nếu như trong chương trình có một số lệnh, hoặc công việc nào đó có dạng giống hoặc tương tự nhau cần lặp đi lặp lại nhiều lần ở các vị trí khác nhau, để khỏi mất thời gian và công sức sao chép những dãy lệnh đó ta nên tổ chức và xây dựng nó thành một đoạn lệnh, chỗ nào trong chương trình cần thực hiện công việc đó thì chỉ cần gọi đoạn lệnh đó mà thôi. Các đoạn lệnh đó gọi là hàm.

Hàm là một đoạn lệnh thực hiện một khối công việc được lặp đi lặp lại nhiều lần trong khi chạy chương trình hoặc dùng tách một khối công việc cụ thể để chương trình đỡ phức tạp.

**Trong ví dụ 1:** Có 2 chương trình, 1 chương trình chính: **main** và 1 chương trình phụ: **loiKhuyen** không có tham số và không trả về giá trị nào nên kiểu dữ liệu của hàm được khai báo là void. Cả main và loiKhuyen đều gọi là hàm. Hàm main là hàm chính thực thi và chạy toàn bộ chương trình của ta. Hàm loiKhuyen thực hiện một số công việc (xuất ra 5 dòng chữ như trên). Trong hàm main ta có 2 chỗ gọi hàm loiKhuyen();

**Trong ví dụ 2**: Cũng tương tự như ví dụ 1, nó gồm 2 hàm, 1 hàm chính là **main** và 1 hàm phụ, nhưng hàm phụ là **vechu(char chu, int soluong)**; có 2 tham số hình thức là **chu** thuộc kiểu char và **soluong** thuộc kiểu int. Hàm vechu cũng không có giá trị trả về nên kiểu dữ liệu của hàm được khai báo là void. Trong hàm main ta có 3 lần gọi hàm vechu với các tham số thực lần lượt như trong chương trình trên.

## 2. Nguyên tắc hoạt động của hàm

* Khi máy gặp lời gọi hàm ở bất kỳ chỗ nào đó thì hàm bắt đầu được thực hiện, tức sẽ rời chỗ đó để đi đến hàm được gọi.
* Nếu là hàm có tham số thì máy sẽ thực hiện truyền tham số thực cho các tham số hình thức tương ứng trong hàm.
* Máy bắt đầu thực hiện lần lượt các câu lệnh trong thân hàm đến khi nào lệnh reuturn hoặc dấu } của hàm thì dừng lại thoát khỏi hàm trở về chương trình đã gọi nó và thực hiện tiếp các câu lệnh của chương trình này.
* Nếu hàm trả về giá trị thì giá trị của biểu thức return trong hàm sẽ là giá trị của hàm.

Lưu ý: Khi khai báo hàm **void vechu(char chu, int soluong)** thì **chu và soluong là các tham số hình thức**, còn khi gọi hàm **vechu(‘h’, 20);** thì **‘h’ và 20 là các tham số thực sự**.

## 3. Cấu trúc của hàm

Chúng ta có cấu trúc khai báo hàm như sau:

|  |  |
| --- | --- |
| 1  2  3  4 | KieuTraVe TenHam(Cac tham so)  {      ThanHam;  } |

Tên hàm được đặt theo quy tắc đặt tên biến.

## 4. Các ví dụ về hàm

### Ví dụ 1: Hàm không trả về giá trị

Ở trên chúng ta đã có 2 ví dụ mở đầu đều là hàm không trả về giá trị và được khai báo có kiểu trả về là **void**.

### Ví dụ 2: Hàm có giá trị trả về

Xây dựng hàm tính n! và tính vài biểu thức giai thừa. Biết n! = 1.2.3…n

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | // e.g about function in C - code by nguyenvanquan7826    #include <stdio.h>    long giaiThua(int n)  {      long gt = 1;      int i;      for (i = 2; i <= n; i++)      {          gt \*= i; // <=> gt = gt \* i      }      return gt;  }    int main()  {      printf("5! = %ld \n", giaiThua(5));      printf("6! + 1 = %ld \n", giaiThua(6) + 1);      return 0;  } |

Kết quả:

5! = 120

6! + 1 = 721

Do n! có thể là một số nguyên khá lớn nên hàm trả về giá trị là kiểu **long** (lớn hơn kiểu int – [Xem lại Các kiểu dữ liệu trong C](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/.2018/01/10/lap-trinh-c-bai-2-cac-thanh-phan-trong-c/#4_Kieu_du_lieu)). Lệnh **return gt** nghĩa là trả về biến gt, gt phải có cùng kiểu dữ liệu với kiểu dữ liệu của hàm (cùng là long).

Trong VD này hàm giaiThua trả về giá trị là giai thừa của số n được truyền vào vì vậy nó có thể được coi như 1 biến và được sử dụng trong các biểu thức, sử dụng để in ra luôn kết quả. Khi gặp hàm ở bất ký đâu thì nó sẽ truyền 1 tham số tương ứng cho hàm và trả về giá trị qua lệnh return.

### Ví dụ 3: Hàm kiểm tra

Xây dựng hàm kiểm tra một số có là số đẹp không? Số đẹp là số chia hết cho 2 và chia hết cho 5.

Để làm bài này, chúng ta cần xác định là hàm này kiểm tra một số là số gì đó, do vậy ta cần trả về giá trị đúng (1) hoặc sai (0). Vậy nên hàm sẽ trả về kiểu số nguyên.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | // e.g about function in C - code by nguyenvanquan7826  #include <stdio.h>    int kiemTraChiaHet(int n)  {      if(n % 2 == 0 && n % 5 == 0)      {          return 1;      }      return 0;  }    int main()  {      int x;        printf("Nhap x = ");      scanf("%d", &x);        if( kiemTraChiaHet(x) == 0)      {          printf("day khong phai so dep @@\n");      } else {          printf("day la so dep !!!\n");      }        return 0;  } |

Ở hàm kiemTraChiaHet, khi n chia hết cho 2 và cho 5, thì sẽ thực hiện lệnh **return 1**, khi thực hiện lệnh này tức là kết thúc hàm, không làm gì trong hàm nãy nữa dù đằng sau còn lệnh hay không. Nếu số n không chia hết cho 2 và cho 5 thì sẽ thực hiện lệnh **return 0**.

### Ví dụ 4: Các hàm gọi nhau

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20 | // e.g about function in C - code by nguyenvanquan7826    #include <stdio.h>    int max2(int a, int b)  {      return a > b ? a : b;  }    int max3(int a, int b, int c)  {      return max2( max2(a, b), c);  }    int main()  {      int a = 7, b = 13, c = 4;      printf("So lon nhat la %d \n", max3(a, b, c));      return 0;  } |

Kết quả:

So lon nhat la 13

Ở ví dụ này chúng ta xây dựng 2 hàm để tìm số lớn nhất của 2 số và 3 số. Trong hàm max2 chúng ta sử dụng [toán tử diều kiện](https://cachhoc.net/2014/12/12/lap-trinh-c-bai-6-ham/2014/12/11/lap-trinh-c-bai-5-cau-truc-re-nhanh-if-else-switch-case-trong-c/#24_Toan_tu_dieu_kien_8211_If_else_rut_gon) để tìm số lớn nhất trong 2 số. Trong hàm max3 ta đã gọi hàm max2 đến 2 lần để tìm số lớn nhất trong a, b rồi lấy số tìm được só sánh với c tìm ra số lớn nhất.

**Chú ý:** Hàm khai báo sau được gọi các hàm đã khai báo trước nó nhưng các hàm khai báo trước không được gọi hàm khai báo sau. Tức là trong VD trên nếu ta xây dựng hàm max2 ở sau hàm max3 thì máy sẽ báo lỗi. Để khắc phục điều này chúng ta thường khai báo các hàm ở đầu chương trình sau đó chúng ta định nghĩa các hàm ở bất kỳ đâu trong chương trình đều được ta xét ví dụ tiếp:

### Ví dụ 5: Khai báo hàm trước khi định nghĩa hàm

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | // e.g about function in C - code by nguyenvanquan7826  #include <stdio.h>    // khai bao cac ham max2, max3, max4  int max2(int a, int b);  int max3(int a, int b, int c);  int max4(int a, int b, int c, int d);    int max3(int a, int b, int c) {      return max2( max2(a, b), c);  }    int max2(int a, int b) {      return a > b ? a : b;  }    int main()  {      int a = 7, b = 13, c = 4, d = 16;      printf("So lon nhat trong a, b, c la %d\n", max3(a, b, c));      printf("So lon nhat trong a, b, c, d la %d\n", max4(a, b, c, d));      return 0;  }    int max4(int a, int b, int c, int d)  {      return max2( max2(a, b), max2(c, d) );  } |

Kết quả:

So lon nhat trong a, b, c la 13

So lon nhat trong a, b, c, d la 16

Khi làm việc chúng ta nên khai báo các hàm trước như thế này và định nghĩa sau. Như vậy chúng ta sẽ tránh được một số lỗi khi mà chúng ta muốn dùng các hàm trong nhau.

Ta đã biết các hàm có thể gọi lẫn nhau và một hàm cũng có thể gọi chính nó, đó là ta sử dụng hàm đệ quy.

### Ví dụ 6: Hàm đệ quy

Ta đã biết công thức tính n! = 1.2.3…n. Tuy nhiên ta cũng có thể viết n! = (n-1)!.n. Trong cách viết này để tính n! chúng ta lại cần tính (n-1)!. Do vậy đây có thể gọi là biểu thức đệ quy và hàm cũng vậy, hàm gọi lại chính nó thì gọi là hàm đệ quy.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | // e.g about function in C - code by nguyenvanquan7826  #include <stdio.h>    int giaiThua(int n)  {      if(n == 0) return 1; // dieu kien dung      return giaiThua(n-1) \* n; // loi goi chinh no  }    int main() {      int n = 5;      printf("%d! = %d",n, giaiThua(n));      return 0;  } |

**Lưu ý** trong hàm đệ quy luôn luôn có 2 yếu tố là điều kiện dừng và lời gọi chính nó.

### Ví dụ 7: Dùng dẫn hướng #define để định nghĩa hàm đơn giản

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10 | #include <stdio.h>    #define tong(x, y) x + y    int main()  {      int a = 5, b = 8;      printf("%d + %d = %d",a, b, tong(a, b));      return 0;  } |

Đến đây chúng ta đã cơ bản nắm được về hàm, nhưng còn rất nhiều điều cần nói về hàm nữa, đặc biệt là hàm liên quan đến mảng và con trỏ. Chúng ta sẽ tiếp tục tìm hiều chúng ở các bài sau cùng với mảng và con trỏ.

**Bài tập**

1. Viết hàm tính tổng S = 1+2+….+n.
2. Viết các hàm kiểm tra số nguyên tố, số hoàn hảo.
3. Nhập vào một dãy n số và thông báo có phải số nguyên tố, số hoàn hảo hay không.

## Lập trình C: Bài 9 – Mảng trong C

**Nội dung**

* [1. Ví dụ mở đầu](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#1_Vi_du_mo_dau)
* [2. Mảng 1 chiều](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#2_Mang_1_chieu)
  + [2.1 Ví dụ tổng quan](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#21_Vi_du_tong_quan)
  + [2.2 Cách khai báo mảng 1 chiều](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#22_Cach_khai_bao_mang_1_chieu)
  + [2.3 Cách truy xuất đến các phần tử trong mảng](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#23_Cach_truy_xuat_den_cac_phan_tu_trong_mang)
  + [2.4 Cách nhập và xuất các phần tử của mảng](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#24_Cach_nhap_va_xuat_cac_phan_tu_cua_mang)
  + [2.5 Nhập xuất mảng sử dụng hàm](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#25_Nhap_xuat_mang_su_dung_ham)
  + [2.6 Một vài lưu ý khác](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#26_Mot_vai_luu_y_khac)
* [3. Mảng 2 chiều](https://cachhoc.net/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/#3_Mang_2_chieu)

Trước khi tìm hiểu về mảng trong C, chúng ta thử làm ví dụ mở đầu.

## 1. Ví dụ mở đầu

Xét ví dụ nhập 5 số nguyên, xuất ra màn hình 5 số đó trên một dòng và tổng 5 số đó.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | // e.g about array - code by nguyenvanquan7826  #include <stdio.h>    int main()  {      int a, b, c, d, e; // 5 bien        printf("Nhap a = ");      scanf("%d", &a);        printf("Nhap b = ");      scanf("%d", &b);        printf("Nhap c = ");      scanf("%d", &c);        printf("Nhap d = ");      scanf("%d", &d);        printf("Nhap e = ");      scanf("%d", &e);        int tong  = a + b + c + d + e;      printf("Tong cua %d + %d + %d + %d + %d = %d \n", a, b, c, d, e, tong);        return 0;  } |

Kết quả:

Nhap a = 3

Nhap b = 5

Nhap c = 3

Nhap d = 6

Nhap e = 1

Tong cua 3 + 5 + 3 + 6 + 1 = 18

Như ví dụ ta thấy nhập hết 5 số và xuất chúng ta màn hình có vẻ rất vất vả và ta đã phải sử dụng quá nhiều biến, nếu số biến của chúng ta không phải là 5 mà là 50, 500,… thì việc này quả là rất khó khăn. Để khắc phục điều này chúng ta hãy sử dụng kiểu mảng.

Mảng Là tập hợp các phần tử có cùng kiểu dữ liệu. Mảng có mảng 1 chiều, 2 chiều,… và mỗi kiểu dữ liệu thì có 1 kiểu mảng tương ứng (mảng nguyên, mảng thực, mảng ký tự (chuỗi)),… Ta chủ yếu xét về mảng 1 chiều và mảng 2 chiều.

Bây giờ chúng ta sẽ giải quyết lại ví dụ trên bằng cách sử dụng mảng 1 chiều.

## 2. Mảng 1 chiều

### 2.1 Ví dụ tổng quan

Các bạn thử đọc và chạy chương trình này, sau đó hãy xem chi tiết ở dưới.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | // e.g about array - code by nguyenvanquan7826  #include <stdio.h>    int main()  {      // khai bao mang a co n phan tu      int n = 5, i, s = 0;      int a[n];        // thuc hien nhap tung phan tu mang      for (i = 0; i < n; i++)      {          printf("Nhap a[%d] = ", i);          scanf("%d", &a[i]);      }        // thuc hien in cac phan tu cua mang ra man hinh      printf("\nMang da nhap \n");      for (i = 0; i < n; i++)      {          printf ("%d \t", a[i]);      }        // tinh tong cac so trong mang      for (i = 0; i < n; i++)      {          s += a[i]; // s = s + a[i]      }      printf("\nTong cac so trong mang: %d\n", s);        return 0;  } |

Nhap a[0] = 3

Nhap a[1] = 4

Nhap a[2] = 6

Nhap a[3] = 2

Nhap a[4] = 7

Mang da nhap

3 4 6 2 7

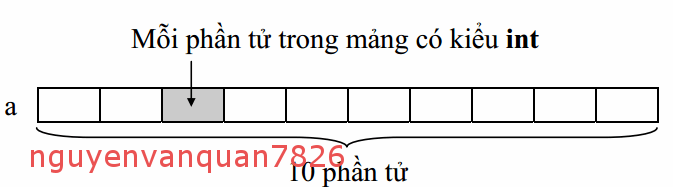
Tong cac so trong mang: 22

### 2.2 Cách khai báo mảng 1 chiều

KieuDuLieu TenMang [SoPhanTu];

VD: int a[10];

Mảng 1 chiều a gồm 10 phần tử thuộc kiểu nguyên. Sau khi khai báo ta có 1 mảng có dạng như sau:

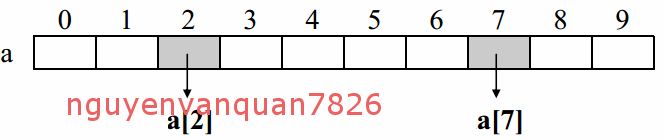


Khai báo mảng a có 10 phần tử

### 2.3 Cách truy xuất đến các phần tử trong mảng

Sau khi mảng được khai báo, mỗi phần tử trong mảng đều có chỉ số để tham chiếu. Chỉ số bắt đầu từ 0 đến n-1 (với n là kích thước mảng). Trong ví dụ trên, ta khai báo mảng 10 phần tử thì chỉ số bắt đầu từ 0 đến 9.

Và ta truy xuất qua cú pháp: TenMang[ChiSo]



Truy xuất mảng trong C

### 2.4 Cách nhập và xuất các phần tử của mảng

Để nhập dữ liệu cho các phần tử trong mảng ta cần duyệt tới từng phần tử trong mảng và tiến hành nhập bằng một vòng for

|  |  |
| --- | --- |
| 1  2  3  4  5 | for (i = 0; i < n; i++)  {      printf("Nhap a[%d] = ", i);      scanf("%d", &a[i]);  } |

Việc xuất các phần tử trong mảng cũng được tiến hành tương tự:

|  |  |
| --- | --- |
| 1  2  3  4 | for (i = 0; i < n; i++)  {      printf ("%d \t", a[i]);  } |

### 2.5 Nhập xuất mảng sử dụng hàm

Trong nhiều bài toán, chúng ta phải nhập nhiều mảng, và xuất nhiều lần, khi đó hãy đưa nhập xuất vào hàm cho dễ dùng.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | // e.g about array - code by nguyenvanquan7826  #include <stdio.h>    void nhapMang(int a[], int n)  {      int i;      // thuc hien nhap tung phan tu mang      for (i = 0; i < n; i++) {          printf("Nhap a[%d] = ", i);          scanf("%d", &a[i]);      }  }    void xuatMang(int a[], int n)  {      int i;      for (i = 0; i < n; i++) {          printf ("%d \t", a[i]);      }  }    int main()  {      // khai bao mang a co n phan tu      int n = 5;      int a[n];      nhapMang(a, n);      xuatMang(a, n);        return 0;  } |

**Lưu ý: Một mảng có mối quan hệ với 1 con trỏ, mảng có thế coi như 1 con trỏ nên khi truyền mảng vào hàm thì các phần tử của mảng sẽ bị ảnh hưởng nếu có sự thay đổi nào trong mảng.**

### 2.6 Một vài lưu ý khác

Khởi tạo mảng: Chúng ta có thể khởi tạo mảng trong quá trình khai báo ngay. VD: float a [5] = {3.4, 5, 6, 7, 4,2}

Với cách khởi tạo này nếu ta khởi tạo vượt quá 5 phần tử máy sẽ báo lỗi, nếu thiếu phần tử nào thì phần tử đó nhận giá trị là 0.

VD: float a [5] = {3.4, 5, 7} => a[4] = a[5] = 0.

VD: float a [5] = {3.4, 5, 7, 1, 2, 3} => máy báo lỗi.

Ngoài ra chúng ta có cách khai báo và khởi tạo mảng chưa biết trước số lượng.

VD: int a[] = {3,6,2,5} => Mảng có 4 phần tử nhận giá trị tương ứng.

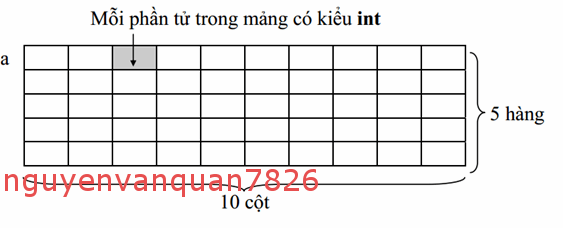
Hoặc: int a[]; => Thường sử dụng khi dùng mảng làm tham số hình thức trong hàm.

## 3. Mảng 2 chiều

Chúng ta có thể coi nó như một ma trận. Các tính chất, cách khai báo, nhập xuất,… cũng tương tự như mảng 1 chiều.

KieuDuLieu TenMang [SoHang][SoCot];

VD khai báo: int a[5][10]; => Mảng a gồm 5 hàng và 10 cột (mỗi hàng có 10 phần tử).

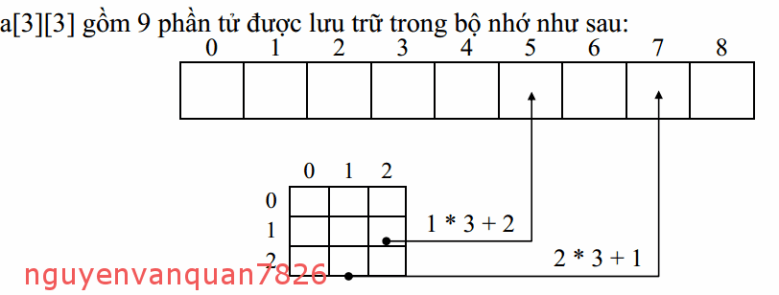


Mảng 2 chiều trong C

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | // e.g about array - code by nguyenvanquan7826  #include <stdio.h>  #define MAX 10    // nhap mang n hang, m cot  void nhapMang(int a[MAX][MAX], int n, int m)  {      int i, j;      for (i = 0; i < n; i++) {          for (j = 0; j < m; j++) {              printf("Nhap a[%d][%d] = ", i, j);              scanf("%d", &a[i][j]);          }      }  }    // xuat mang n hang, m cot  void xuatMang(int a[MAX][MAX], int n, int m)  {      int i, j;      for (i = 0; i < n; i++) {          for (j = 0; j < m; j++) {              printf ("%-3d", a[i][j]);          }          printf("\n"); // xuong dong khi het 1 dong      }  }    int main()  {      // khai bao mang a co n phan tu      int n = 2, m = 3;      int a[MAX][MAX];      nhapMang(a, n, m);      xuatMang(a, n, m);        return 0;  } |

**Chú ý:** Thực chất trong bộ nhớ, các phần tử của mảng 2 chiều (và mảng nhiều chiều) được tổ chức như là nhiều mảng 1 chiều liên tiếp nhau, vì vậy ta có thể quy mảng 2 chiều thành mảng 1 chiều.

chap9-luu-tru-mang-2-chieu-trong-c.png



Cách tổ chức, lưu trữ mảng 2 chiều trong C

**Bài tập**

1. Nhập vào một mảng, tìm số lớn nhất và nhỏ nhất trong mảng đã nhập.
2. Nhập vào một dãy số, in ra các vị trí số lớn nhất (có thể có nhiều vị trí).
3. Nhập vào một dãy số là lượng mưa của các tháng liên tiếp trong năm. Thông báo ra các tháng có lượng mưa lớn hơn lượng mưa trung bình của các tháng đó.
4. Nhập vào một dãy số, sắp xếp và in ra dãy số theo chiều tăng dần.
5. Nhập vào một ma trận (mảng 2 chiều). In ra tổng của mỗi hàng trong ma trận đó.

## Lập trình C: Bài 10 – Chuỗi ký tự trong C

**Nội dung**

* [1. Tổng quan](https://cachhoc.net/2014/12/16/lap-trinh-c-bai-8-chuoi-ky-tu-trong-c/#1_Tong_quan)
* [2. Một số cách khai báo, khởi tạo chuỗi](https://cachhoc.net/2014/12/16/lap-trinh-c-bai-8-chuoi-ky-tu-trong-c/#2_Mot_so_cach_khai_bao_khoi_tao_chuoi)
* [3. Một số ví dụ về chuỗi ký tự trong C](https://cachhoc.net/2014/12/16/lap-trinh-c-bai-8-chuoi-ky-tu-trong-c/#3_Mot_so_vi_du_ve_chuoi_ky_tu_trong_C)
  + [3.1 Ví dụ 1: Đếm số từ trong chuỗi](https://cachhoc.net/2014/12/16/lap-trinh-c-bai-8-chuoi-ky-tu-trong-c/#31_Vi_du_1_Dem_so_tu_trong_chuoi)
  + [3.2 Ví dụ 2: Chuẩn hóa chuỗi](https://cachhoc.net/2014/12/16/lap-trinh-c-bai-8-chuoi-ky-tu-trong-c/#32_Vi_du_2_Chuan_hoa_chuoi)
* [4. Một số hàm về chuỗi và ký tự](https://cachhoc.net/2014/12/16/lap-trinh-c-bai-8-chuoi-ky-tu-trong-c/#4_Mot_so_ham_ve_chuoi_va_ky_tu)

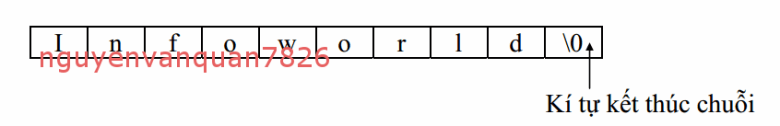
Ở bài [Nhập xuất trong c](https://cachhoc.net/2014/12/16/lap-trinh-c-bai-8-chuoi-ky-tu-trong-c/2018/01/11/lap-trinh-c-bai-3-nhap-xuat-trong-c/) chúng ta đã làm quen với cách khai báo, nhập xuất chuỗi ký tự trong C một cách đơn giản. Trong bài này chúng ta sẽ nói nhiều hơn về các ví dụ, bài tập về chuỗi .

## 1. Tổng quan

Chuỗi được xem như là một mảng 1 chiều gồm các phần tử có kiểu char như ký tự, con số và bất cứ ký tự đặc biệt như +, -, \*, /, $, #,…

Theo quy ước, một chuỗi sẽ được kết thúc bởi ký tự null (‘\0’ : kí tựrỗng).

Ví dụ: chuỗi “Infoworld” được lưu trữ như sau:



Cách lưu trữ chuỗi ký tự

## 2. Một số cách khai báo, khởi tạo chuỗi

Chúng ta xét ví dụ sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | // e.g about string - code by nguyenvanquan7826  #include <stdio.h>    int main()  {      // khai bao chuoi co toi da 50 ky tu      char name[50];      printf("Hi, What is your name? \nMy name is: ");      gets(name);      printf("Hi %s, welcome to C language\n", name);        // khoi tao chuoi ngay khi khai bao      char myLove[] = "Nguyen Thi Lap Lanh";      puts(myLove);        return 0;  } |

Kết quả:

Hi, What is your name?

My name is: Nguyen Van Quan

Hi Nguyen Van Quan, welcome to C language

Nguyen Thi Lap Lanh

Trong chương trình trên, mình có dùng hàm **puts** để in chuỗi myLove ra, đây cũng là hàm để xuất chuỗi.

Như trên chúng ta có thể thấy là khai báo chuỗi sau đó nhập chuỗi hoặc vừa khai báo vừa gán giá trị cho chuỗi ngay. Tuy nhiên chúng ta không thể khai báo sau đó mới gán giá trị như sau:

|  |  |
| --- | --- |
| 1  2 | char name[50];  name = "Nguyen Van Quan"; // error |

Trong trường muốn khai báo sau đó mới gán giá trị, chúng ta phải dùng hàm copy chuỗi **strcpy** nằm trong thư viện **string.h** để lưu giá trị như sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | // e.g about string - code by nguyenvanquan7826  #include <stdio.h>  #include <string.h>    int main()  {      char name[50];      strcpy(name, "Nguyen Van Quan");      puts(name);        return 0;  } |

## 3. Một số ví dụ về chuỗi ký tự trong C

### 3.1 Ví dụ 1: Đếm số từ trong chuỗi

Hãy nhập vào một chuỗi ký tự và đếm số từ trong chuỗi. VD chuỗi “Nguyen Van Quan” có 3 từ.

Để làm bài này, chúng ta thấy mỗi từ là các ký tự liên tiếp nhau và phân tách các từ bằng các dấu cách. Do vậy bài này sẽ quy về đếm số dấu cách trong chuỗi. Nếu chuỗi có 1 từ thì không có dấu cách, chuỗi có 2 từ thì có 1 dấu cách giữa 2 từ đó. Tổng quát là có n từ thì sẽ có n-1 dấu cách.

Vấn đề tiếp theo là làm sao đếm được các dấu cách? Đơn giản, như mở đầu ta đã biết chuỗi là một mảng các ký tự, do vậy chúng ta có thể duyệt lần lượt các ký tự của chuỗi để kiếm tra ký tự nào là dấu cách. Nhưng muốn duyệt hết các ký trong chuỗi (mảng ký tự) thì phải biết số lượng ký tự có trong chuỗi (số lượng phần tử có trong mảng).

May mắn là trong thư viện **string.h** chúng ta có một hàm để lấy độ dài của chuỗi là hàm **strlen**. (str – string, len – length).

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | // e.g about string - code by nguyenvanquan7826  #include <stdio.h>  #include <string.h> // for strlen function    int main()  {      char s[50];      printf("Enter a string: ");      gets(s);        int i, count = 0; // count - bien dem so luong dau cach      for (i = 0; i < strlen(s); i++ )      {          if(s[i] == ' ')          {              count++;          }      }        printf("Number word in string is: %d\n", count + 1 );        return 0;  } |

Code khá đơn giản, các bạn đọc, hiểu và chạy thử nhé.

Tuy nhiên code trên chúng ta có một số lưu ý:

* Để biểu diễn ký tự thì ta đặt trong cặp nháy đơn, chuỗi thì chúng ta đặt trong cặp nháy kép. Nên ở trên dấu cách đặt trong cặp nháy đơn và chúng ta có thể so sánh 2 ký tự bằng các phép so sánh như với 2 số, còn 2 chuỗi thì không thể so sánh được như vậy, các bạn có thể đọc thêm về [cách so sánh chuỗi](https://cachhoc.net/2014/12/16/lap-trinh-c-bai-8-chuoi-ky-tu-trong-c/2018/01/19/sanh-chuoi-trong-c-strcmp/).
* Do chuỗi là một mảng các ký tự, nên muốn lấy ký tự thứ i trong chuỗi s thì ta truy cập như với mảng là s[i].
* Ví dụ này chỉ áp dụng khi chuỗi có độ dài lớn hơn 0 và không có dấu trắng thừa ở đầu, cuối hoặc giữa các từ.
* Như vòng lặp for ở trên, chúng ta có điều kiện i < strlen(s), tuy nhiên bản chất của hàm strlen là một vòng lặp nữa để đếm số lượng ký tự của chuỗi s. Nên nếu chúng ta viết trực tiếp điều kiện như trên thì trong mỗi lần lặp, chương trình lại phải chạy lại lệnh strlen lặp để đếm số lượng ký tự của s. Điều này là thừa và làm chương trình chạy lâu hơn. Do vậy chúng ta sẽ đặt 1 biến là độ dài của chuỗi ra ngoài như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | int len = strlen(s);  for (i = 0; i < len; i++ )  {      if(s[i] == ' ')      {          count++;      }  } |

### 3.2 Ví dụ 2: Chuẩn hóa chuỗi

Hãy nhập vào một chuỗi và xóa bỏ toàn bộ các dấu cách thừa ở đầu, cuối và giữa các từ nếu có.

Bài toán này là bài toán quan trọng cho các phần mềm, sau này khi lưu trữ, nhập liệu cần lưu ý để dữ liệu được chuẩn, không thừa thiếu gây sai sót trong quá trình xử lý và tím kiếm.

* Ký tự đầu tiên của chuỗi s là dấu cách thì s[0] là dấu cách, chúng ta xóa nó là xong.
* Các ký tự cách giữa các từ nếu thừa tức là s[i] và s[i+1] cùng là dấu cách. Chúng ta xóa 1 trong 2 là ok, vì các từ sẽ phân tách nhau bởi 1 dấu cách nên ta phải dữ lại 1 dấu cách.
* Các ký tự ở cuối chuỗi là dấu cách thì chúng ta sẽ xóa bằng cách gán ký tự cuối cùng là ký tự rỗng '\0' là xong. Nhớ rằng ký tự cuối cùng của mảng n phần tử là a[n-1], do vậy ký tự của chuỗi là s[ strlen(s) - 1 ].

Vấn đề tiếp theo là làm sao xóa 1 ký tự trong chuỗi? Các bạn nhìn lại ví dụ trên chúng ta có dùng hàm copy chuỗi, và để xóa 1 hoặc một số ký tự trong chuỗi chúng ta sẽ dùng hàm này nhưng sẽ dùng theo copy địa chỉ.

Để xóa từ ký tự i đến ký tự j trong chuỗi s, chúng ta dùng lệnh strcpy(&s[i], &s[j+1]);. Bản chất là chúng ta copy địa chỉ của s[j+1] về địa chỉ của s[i].

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | // e.g about string - code by nguyenvanquan7826  #include <stdio.h>  #include <string.h>    int main()  {      char s[50];      printf("Enter a string: ");      gets(s);        // delete all space at start of string      while( s[0] == ' ' ) strcpy(&s[0], &s[1]);        // delete all space at end of string      while( s[ strlen(s)-1 ] == ' ') s[ strlen(s)-1 ] = '\0';        // delete all space between two word        int i;      for(i = 0; i < strlen(s); i++)      {          if( s[i] == ' ' && s[i+1] == ' ')          {              strcpy(&s[i], &s[i+1]);              i--; // why???          }      }        printf("s=%s.\n", s);        return 0;  } |

Ok. Các bạn chạy thử nhé. Nhớ nhập chuỗi thừa các dấu cách ở đầu, cuối, giữa để kiểm tra.

Một câu hỏi nhỏ coi như bài tập làm thêm cho các bạn là hãy nhìn dòng code i--; // why??? và nghĩ tại sao lại có dòng này? Tại sao i phải giảm đi 1?

## 4. Một số hàm về chuỗi và ký tự

**Các hàm kiểm tra ký tự**. (các hàm này trong thư viện **ctype.h**) Nếu đúng thì hàm cho giá trị khác 0. Nếu sai thì hàm cho giá trị bằng 0.

* Int isalpha(int c) : kiểm tra ký tự có là chữ cái không.
* Int isdigit(int c) : kiểm tra xem ký tự có là chữ số không.
* Int islower(int c): kiểm tra ký tự có là chữ thường không.
* Int isupper(int c): kiểm tra ký tự có là chữ hoa không.
* Int ispace(int c): kiểm tra ký tự có là trống không (\n, dấu cách, \t).

**Các hàm sử lý xâu ký tự**. (các hàm này nằm trong thư viện **string.h**)

* Int strlen(char \*s) trả về độ dài của xâu s;
* Char \*strupr(char \*s) đổi chữ thường trong xâu s sang chữ hoa.
* Char \*strlwr(char \*s) đổi chữ hoa sang chữ thường.
* Char \*strcat(char \*s1, char \*s2) nối xâu s2 vào xâu s1;
* Int strcmp(char \*s1, char \*s2) cho giá trị âm nếu xâu s1 nhở hơn xâu s2. Và cho giá tị dương nếu xâu s1 lớn hơn xâu s2. Trả về giá trị bằng 0 nếu xâu s1 bằng xâu s2.
* Int strcmpi (char \*s1, char \*s2) so sánh 2 xâu nhưng không phân biệt chữ thường và chữ hoa.
* Char \*strcpy(char \*s1, char \*s2) copy xâu s2 vào xâu s1.
* Char \*strncpy(char \*s1, char \*s2, int n) sao chép n ký tự đầu của xâu s2 sang xâu s1
* Char \*strnset(char \*s ,int c, int n) dùng để sao chép n lần ký tự c vào xâu s.
* Char \*strstr(char \*s1, char \*s2) tìm sự xuất hiện của xâu s2 trong xâu s1. Nếu tìm thấy hàm cho địa chỉ của xâu con trong xâu s1. Trái lại cho NULL.
* Char \*strrev(char \*s) dùng đảo ngược xâu s.Nếu thành công hàm cho địa chỉ xâu đã đảo.

**Bài tập**

1. Viết chương trình tách tên từ một chuỗi cho trước. VD tên Nguyen Thi Lap Lanh => tách được Lanh
2. Viết chương trình chuẩn hóa chuỗi tên riêng. VD:ha noi => Ha Noi.
3. Viết hàm chuyển đổi 1 chuỗi sang chữ thường và 1 hàm chuyển đổi sang chữ HOA.
4. Viết chương trình nhập vào một chuỗi ký tự rồi đếm xem trong chuỗi đó có bao nhiêu chữ “ng”.
5. Viết chương trình nhập vào một chuỗi ký tự. Kiểm tra xem chuỗi đó có đối xứng không? Chuỗi đối xứng là chuỗi khi viết ngược lại vẫn được như chuỗi ban đầu. VD **level**
6. Viết chương trình nhập vào số có 3 chữ số. Cho biết dòng chữ mô tả giá trị con số đó. Ví dụ 123 -> một trăm hai mươi ba.

**Lập trình C: Bài 11 – Con trỏ trong C**

**Nội dung**

* [1. GIới thiệu con trỏ trong C](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#1_GIoi_thieu_con_tro_trong_C)
* [2. Biến con trỏ](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#2_Bien_con_tro)
  + [2.1 Ví dụ](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#21_Vi_du)
  + [a. Khai báo biến con trỏ](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#a_Khai_bao_bien_con_tro)
  + [b. Quy định vùng trỏ tới của con trỏ](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#b_Quy_dinh_vung_tro_toi_cua_con_tro)
  + [c. Cách truy xuất](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#c_Cach_truy_xuat)
  + [d. Một số phép toán trên biến con trỏ](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#d_Mot_so_phep_toan_tren_bien_con_tro)
* [3. Cấp phát và thu hồi vùng nhớ](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#3_Cap_phat_va_thu_hoi_vung_nho)
  + [a. Cấp phát:](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#a_Cap_phat)
  + [b. Thu hồi và kiểm tra vùng nhớ còn lại](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#b_Thu_hoi_va_kiem_tra_vung_nho_con_lai)
* [3. Hàm có đối là con trỏ](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/#3_Ham_co_doi_la_con_tro)

**1. Giới thiệu con trỏ trong C**

Các biến chúng ta đã biết và sử dụng trước đây đều là biến có kích thước và kiểu dữ liệu xác định. Người ta gọi các biến kiểu này là biến tĩnh. Khi khai báo biến tĩnh, một lượng ô nhớ cho các biến này sẽ được cấp phát mà không cần biết trong quá trình thực thi chương trình có sử dụng hết lượng ô nhớ này hay không. Mặt khác, các biến tĩnh dạng này sẽ tồn tại trong suốt thời gian thực thi chương trình dù có những biến mà chương trình chỉ sử dụng 1 lần rồi bỏ.

Một số hạn chế có thể gặp phải khi sử dụng các biến tĩnh:

* Cấp phát ô nhớ dư, gây ra lãng phí ô nhớ.
* Cấp phát ô nhớ thiếu, chương trình thực thi bị lỗi.

Để tránh những hạn chế trên, ngôn ngữ C cung cấp cho ta một loại biến đặc biệt gọi là biến động với các đặc điểm sau:

* Chỉ phát sinh trong quá trình thực hiện chương trình chứ không phát sinh lúc bắt đầu chương trình.
* Khi chạy chương trình, kích thước của biến, vùng nhớ và địa chỉ vùng nhớ được cấp phát cho biến có thể thay đổi.
* Sau khi sử dụng xong có thể giải phóng để tiết kiệm chỗ trong bộ nhớ. Tuy nhiên các biến động không có địa chỉ nhất định nên ta không thể truy cập đến chúng được. Vì thế, ngôn ngữ C lại cung cấp cho ta một loại biến đặc biệt nữa để khắc phục tình trạng này, đó là biến con trỏ (pointer) với các đặc điểm:
* Biến con trỏ không chứa dữ liệu mà chỉ chứa địa chỉ của dữ liệu hay chứa địa chỉ của ô nhớ chứa dữ liệu.
* Kích thước của biến con trỏ không phụ thuộc vào kiểu dữ liệu, luôn có kích thước cố định là 2 byte.

**2. Biến con trỏ**

Mỗi biến khi được khai báo đều được cấp phát cho 1 vùng nhớ nhất định ở những nơi (địa chỉ) khác nhau. Biến con trỏ **là biến** dùng để lưu trữ địa chỉ của các biến đó.

**2.1 Ví dụ**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | // e.g about pointer - code by nguyenvanquan7826  #include <stdio.h>    int main()  {      /\* khai bao bien x va bien con tro px \*/      int x, \*px;      px = &x;      /\* &x : tra ve dia chi cua bien x       \* px = &x : gan dia chi cua bien x cho px hay px tro den x       \*/        x = 42;        printf("Vi tri cua bien x la %p \n", &x);      printf("Noi dung cua bien x la %d \n", x);      printf("Vi tri cua bien x la %p \n", px);      printf("Noi dung cua bien x la %d \n", \*px);        \*px = 7826;      printf("\n -------- \n\n");        printf("Noi dung cua bien x la %d \n", x);      printf("Noi dung cua bien x la %d \n", \*px);        return 0;  } |

Kết quả:

Kết quả:

Noi dung cua bien x la 42

Vi tri cua bien x la 0x7ffe064348fc

Noi dung cua bien x la 42

--------

Noi dung cua bien x la 7826

Noi dung cua bien x la 7826

Qua ví dụ mở đầu này ta có thể rút ra một số điểm sau:

**a. Khai báo biến con trỏ**

Với mỗi kiểu dữ liệu ta có tương ứng một biến con trỏ có kiểu đó.

**Kiểu \* Tên biến con trỏ;**

Trong VD trên ta khai báo 1 biến con trỏ px thuộc kiểu int.

**b. Quy định vùng trỏ tới của con trỏ**

Ta dùng toán tử & để lấy địa chỉ của 1 biến và sau đó gán địa chỉ đó cho biến con trỏ.

**Tên con trỏ = &biến;**

**c. Cách truy xuất**

Với con trỏ px bên trên ta có 2 phép tuy xuất là:

* px : Lấy địa chỉ mà nó lưu giữ (trỏ tới)
* \*px : Lấy giá trị trong vùng nhớ mà nó trỏ tới.

Trong VD trên ta có thể thấy sau phép gán **px = &x;** thì việc ta viết:

* px sẽ tương đương với &x
* \*px sẽ tương đương với x. và ta có thể sử dụng \*px trong các phép toán, biểu thức.

**d. Một số phép toán trên biến con trỏ**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | // e.g about pointer - code by nguyenvanquan7826  #include <stdio.h>    int main()  {      /\* khai bao bien x va 2 bien con tro px, qx \*/      int x, \*px, \*qx;      px = &x;        printf("Nhap gia tri cho vung nho px tro toi: ");      scanf("%d", px);      /\* px la con tro nen khong viet scanf("%d", &px);  \*/        qx = px; /\* gan gia tri cua px cho qx, qx cun tro toi x\*/        printf("Vi tri cua bien x la %p \n", &x);      printf("Vi tri cua bien x la %p \n", px);      printf("Vi tri cua bien x la %p \n", qx);      printf("Noi dung cua bien x la %d \n", x);      printf("Noi dung cua bien x la %d \n", \*px);      printf("Noi dung cua bien x la %d \n", \*qx);        // tang gia tri cua o nho len, <=> x = x + 7826      \*px += 7826;      printf("Noi dung cua bien x la %d \n", x);        px++;      /\* cong them mot don vi cho px       \*  => px tro toi vung nho tiep theo       \*/        printf("Vi tri px tro toi la %p \n", px);        return 0;  } |

Kết quả:

Nhap gia tri cho vung nho px tro toi: 42

Vi tri cua bien x la 0xbfba58a0

Vi tri cua bien x la 0xbfba58a0

Vi tri cua bien x la 0xbfba58a0

Noi dung cua bien x la 42

Noi dung cua bien x la 42

Noi dung cua bien x la 42

Noi dung cua bien x la 7868

Vi tri px tro toi la 0xbfba58a4

Trong vd trên ta thấy có một số phép toán trên con trỏ hay gặp sau: (ngoài ra còn nhiều phép toán khác).

* 2 biến con trỏ cùng kiểu có thể được gán cho nhau hoặc thực hiện các phép toán cộng cho một số nguyên, trừ 2 con trỏ cho nhau. Ở VD trên ta thực hiện các phép toán:
* Gán: qx = px; Khi này qx nhận giá trị của px hiện có là địa chỉ của biến x, tức là qx và px cùng trỏ đến x. ngoài ra ta có thể gán như sau: qx = px + 2; với qx, px là các biến con trỏ cùng kiểu. Phép trừ 2 con trỏ cùng kiểu sẽ trả về 1 giá trị nguyên (int). Đây chính là khoảng cách (số phần tử) giữa 2 con trỏ đó
* Tăng: Các phép tăng giảm, cộng trừ được thực hiện trên biến con trỏ tương tự như với các biến số học. Điểm khác biệt duy nhất là nó tăng giảm, cộng trừ theo đơn byte mà kiểu của nó có.
* VD. Trong VD trên ta có phép tăng: px++; Khi này giả sử px đang trỏ đến địa chỉ: 0xbfba58a0 thì sau phép tăng nó có giá trị là (trỏ đến vị trí) 0xbfba58a4 (tăng lên 4) vì px là con trỏ kiểu int mà mỗi biến kiểu int chiếm 4 byte trong bộ nhớ.
* Ngoài ra chúng ta để ý còn phép thay đổi giá trị của biến x bằng phép toán \*px += 3; Ở phép toán này thực chất là ta đã thay đổi giá trị ở ô nhớ (địa chỉ) mà px trỏ tới, từ đó dẫn đến giá trị của biến x cũng thay đổi theo.

**Các bạn chú ý:**

* Tùy theo trình dịch mà dung lượng của các kiểu là khác nhau. (trong trình dịch này thì kiểu int chiếm 4 byte nhưng trong trình dịch khác thì nó lại chiếm 2 byte). Để biết dung lượng từng kiểu bạn dùng toán tử sizeof() mà trong bài 2 đã đề cập.
* Mỗi biến con trỏ, dù là con trỏ thuộc kiểu nào (int, float, double,…) cũng chỉ chiếm 2 byte bộ nhớ để lưu trữ đĩa chỉ của các biến.

**3. Cấp phát và thu hồi vùng nhớ**

**a. Cấp phát:**

Trước khi vào phần này ta làm ví dụ nho nhỏ.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | // e.g about pointer - code by nguyenvanquan7826  #include <stdio.h>    int main()  {      int \*px;      \*px = 42;      printf("Vi tri con tro px la %p \n", px);      printf("Gia tri con tro px tro toi la %d \n", \*px);      return 0;  } |

Khi biên dịch thì sẽ không co lỗi (có cảnh báo), khi chạy sẽ không thể chạy được mà chương trình sẽ thoát ra luôn.

Nguyên nhân là khi khai báo biến con trỏ px thì máy mới chỉ cung cấp 2 byte để lưu địa chỉ của biến con trỏ mà chưa cấp phát vùng nhớ để con trỏ px lưu trữ dữ liệu. (tương tự như hợp tác xã cung cấp 2 Kg thóc cho bạn để làm giống nhưng lại không cung cấp cho bạn ruộng đất để bạn reo mạ vậy ).

**Lưu ý:** Có một số trình dịch sẽ không báo lỗi mà vẫn chạy bình thường nhưng tốt nhất là ta nên cấp phát trước khi sử dụng. Lỗi này sẽ xuất hiện rõ nhất khi bạn sử dụng con trỏ với mảng mà lát nữa ta sẽ đề cập.

Thôi ta đi vào vấn đề chính, làm sao để cấp phát vùng nhớ cho con trỏ.

Để cấp phát vùng nhớ cho con trỏ ta dùng các hàm sau trong thư viện stdlib.h.

* malloc : tên con trỏ = (kiểu con trỏ \*) malloc (sizeof(kiểu con trỏ));
* calloc : tên con trỏ = (kiểu con trỏ \*) malloc (n, sizeof(kiểu con trỏ));

Trong đó sizeof(kiểu con trỏ) là kích thước của kiểu; n là số lần của sizeof(kiểu con trỏ) được cấp.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | // e.g about pointer - code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    int main()  {      int \*px, \*qx;      px = (int \*) malloc(sizeof(int));      qx = (int \*) calloc(1, sizeof(int));        printf("Vi tri con tro px la %p \n", px);      printf("Gia tri con tro px tro toi la %d \n", \*px);        printf("Vi tri con tro qx la %p \n", qx);      printf("Gia tri con tro qx tro toi la %d \n", \*qx);      return 0;  } |

Ở đây các bạn chú ý: sự khác nhau duy nhất giữa malloc và calloc mà các bạn hiểu đơn giản là với malloc thì khi cấp phát máy sẽ cấp phát cho px 1 ô bất kỳ mà không cần biết ô đó có dữ liệu là gì hay không có dữ liệu (do đó \*px có giá trị như trên) còn calloc cũng vậy nhưng khác 1 điểm là sau khi cấp phát thì máy sẽ tự động gán luôn giá trị 0 cho ô nhớ mà biến qx trỏ tới, tức qx có giá trị mặc định là 0.

Khi cấp phát cho biến con trỏ 1 số lượng ô nhớ nào đó mà trong quá trình làm việc ta thiếu và cần cấp phát thêm thì ta sử dụng lệnh **realloc**:

**tên con trỏ = (kiểu con trỏ \*) realloc (tên con trỏ, số lượng cần cấp phát \* sizeof(kiểu con trỏ));**

Trong đó: số lượng cần cấp phát = cũ + mới.

VD: Ban đầu ta cấp phát cho con trỏ px là 10 ô nhớ.

Sau đó muốn cấp phát thêm cho nó 5 ô nhớ nữa thì số lượng cấp phát = 15.

**b. Thu hồi và kiểm tra vùng nhớ còn lại**

Để thu hổi bộ nhớ đã cấp phát ta dùng hàm **free(tên con trỏ);**

**4. Hàm có đối là con trỏ**

Như trong bài [Hàm hoán vị trong C](https://cachhoc.net/2014/12/17/lap-trinh-c-bai-9-con-tro-trong-c/2018/01/20/ham-hoan-vi-trong-c/) chúng ta đã biết cách truyền các tham số a,b trong hàm HoanVi là cách truyền bằng tham trị chứ không phải truyền bằng địa chỉ (hay tham biến) nên mặc dù trong hàm thì giá trị các biến đã được thay đổi nhưng sau khi hàm thực hiện xong thì các giá trị vẫn chưa thể thay đổi được. Và ta sẽ phải sửa lại bằng cách truyền tham số hình thức là con trỏ a và con trỏ b để khi thực hiện hoán đổi có thể hoán đổi tại địa chỉ của các ô nhớ đó. Khi đó ta mới có được kết quả mong muốn.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | // e.g about pointer - code by nguyenvanquan7826  #include <stdio.h>    void hoanVi(int \*a, int \*b)  {       int temp = \*a;       \*a = \*b;       \*b = temp;  }    int main()  {      int a = 42, b = 7826;      printf("Truoc khi goi ham hoan vi: a = %d, b = %d \n", a, b);        hoanVi(&a, &b);        printf("Sau khi goi ham hoan vi: a = %d, b = %d \n", a, b);        return 0; |

**Lập trình C: Bài 12 – Mối quan hệ giữa con trỏ và mảng, chuỗi ký tự**

**Nội dung**

* [1. Con trỏ và mảng 1 chiều.](https://cachhoc.net/2014/12/18/lap-trinh-c-bai-10-moi-quan-giua-con-tro-va-mang-chuoi-ky-tu/#1_Con_tro_va_mang_1_chieu)
* [2. Nhập mảng trong hàm](https://cachhoc.net/2014/12/18/lap-trinh-c-bai-10-moi-quan-giua-con-tro-va-mang-chuoi-ky-tu/#2_Nhap_mang_trong_ham)
* [3.Con trỏ và xâu ký tự](https://cachhoc.net/2014/12/18/lap-trinh-c-bai-10-moi-quan-giua-con-tro-va-mang-chuoi-ky-tu/#3Con_tro_va_xau_ky_tu)
* [4. Con trỏ và mảng 2 chiều, mảng các con trỏ – Con trỏ đa cấp](https://cachhoc.net/2014/12/18/lap-trinh-c-bai-10-moi-quan-giua-con-tro-va-mang-chuoi-ky-tu/#4_Con_tro_va_mang_2_chieu_mang_cac_con_tro_Con_tro_da_cap)
  + [a. Con trỏ và mảng 2 chiều](https://cachhoc.net/2014/12/18/lap-trinh-c-bai-10-moi-quan-giua-con-tro-va-mang-chuoi-ky-tu/#a_Con_tro_va_mang_2_chieu)
  + [b. Mảng các con trỏ](https://cachhoc.net/2014/12/18/lap-trinh-c-bai-10-moi-quan-giua-con-tro-va-mang-chuoi-ky-tu/#b_Mang_cac_con_tro)

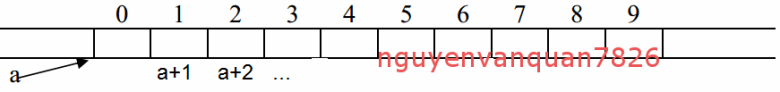
**1. Con trỏ và mảng 1 chiều.**

Như trong bài [Mảng](https://cachhoc.net/2014/12/18/lap-trinh-c-bai-10-moi-quan-giua-con-tro-va-mang-chuoi-ky-tu/2014/12/13/lap-trinh-c-bai-7-mang-trong-c/) chúng ta đã biết ta có thể coi biến mảng như một con trỏ, vì vậy ta có thể sử dụng chính biến mảng đó để truy cập mảng theo cách của con trỏ.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | //code by nguyenvanquan7826  #include <stdio.h>    void nhapMang(int a[], int n)  {      int i;      for (i = 0; i < n; i++)      {          printf("Nhap a[%d] = ", i);          scanf("%d", &a[i]);      }  }    void nhapContro(int a[], int n)  {      int i;      for (i = 0; i < n; i++)      {          printf("Nhap a[%d] = ", i);          scanf("%d", a + i);      }  }    void xuatMang(int a[], int n)  {      int i;      for (i = 0; i < n; i++)      {          printf ("%d \t", a[i]);      }  }    int main()  {      // khai bao mang a co n phan tu      int n = 5;      int a[n];      nhapContro(a, n);      xuatMang(a, n);        return 0;  } |

Ở hàm thứ nhất ta đã quen với cách nhập bình thường, mình sẽ không nói nhiều nữa.

Ở hàm thứ hai, chúng ta chỉ thay &a bằng a+i vì khi khai báo a[20] thì a coi như là một con trỏ và máy sẽ cấp phát cho ta các ô nhớ liên tiếp từ a đến a + 19. Và a + i là địa chỉ của a[i] (tức nó tương đương với &a[i]). a trỏ đến vị trí a[0].



Con trỏ và mảng trong C

Ngoài ra bạn có thể khai báo 1 mảng sau đó dùng 1 con trỏ trỏ tới đầu mảng thì con trỏ đó cũng trở thành mảng đó.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | //code by nguyenvanquan7826  #include <stdio.h>    int main()  {      int n = 5, i;      int a[n], \*pa;      pa = a; // con tro pa tro toi dau mang a        for (i = 0; i < n; i++)      {          printf("Nhap a[%d] = ", i);          scanf("%d", pa + i);      }        for (i = 0; i < n; i++)      {          printf ("%d \t", \*(pa + i));      }        return 0;  } |

Các bạn chú ý: tại sao ta không cần cấp phát ô nhớ cho con trỏ pa mà vẫn sử dụng được bình thường, bởi vì ta đã khai báo mảng a[20] nên máy đã cấp phát ô nhớ để lưu trữ mảng a, khi ta thực hiện trỏ con trỏ pa tới mảng a thì các ô nhớ này đã có rồi nên không cần cấp phát ô nhớ cho pa nữa. Ta xét ví dụ sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    int main()  {      int n = 5, i;        // cap phat bo nho cho pa      int \*pa = (int \*) malloc(n \* sizeof(int));        for (i = 0; i < n; i++)      {          printf("Nhap a[%d] = ", i);          scanf("%d", pa + i);      }        for (i = 0; i < n; i++)      {          printf ("%d \t", \*(pa + i));      }        return 0;  } |

Ở trong VD này ta không khai báo mảng a như VD trước, do đó cũng không thể trỏ pa tới vị trí nào, mà muốn thực hiện được ta cần cấp phát các ô nhớ cho pa như trên.

**2. Nhập mảng trong hàm**

Việc nhập mảng không phải lúc nào cũng thuận lợi vì như các ví dụ trước chúng ta cần phải có số lượng phần tử của mảng trước khi cấp phát hoặc nhập, vậy nếu chúng ta chưa biết trước số phần tử mà lại phải dùng hàm để nhập mảng thì sao. Các bạn xem ví dụ sau.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    void nhapContro(int \*(\*a), int \*n)  {      int i;        printf("Nhap so phan tu cua mang: ");      scanf("%d", n); // khong phai &n      \*a = (int \*) malloc ((\*n) \* sizeof(int));      // \*a : lay dia chi cua mang a chu khong phai gia tri cua a        for (i = 0; i < \*n; i++)      {          printf("Nhap a[%d] = ", i);          scanf("%d", (\*a + i));      }  }    void xuatMang(int \*a, int n)  {      int i;      for (i = 0; i < n; i++)      {          printf ("%d \t", a[i]);      }  }    int main()  {      int \*a, n;        nhapContro(&a, &n); // lay dia chi cua a va n      xuatMang(a, n);        return 0;  } |

Trong VD này ta thực hiện nhập và xuất mảng trong hàm, cấp phát bộ nhớ cũng trong hàm luôn. Các chú thích mình đã ghi rõ trong chương trình rồi.

Có 1 điểm chú ý là trong hàm nhập mảng a bằng con trỏ thì có 2 dấu \*. 1 dấu là của mảng a (dấu thứ 2), còn dấu đầu tiên là dùng để truyền địa chỉ làm giá trị của mảng có thể giữ nguyên khi ra khỏi hàm, nó giống như là dấu \* trong hàm **HoanVi(int \*a,int \*b)** vậy.

**3.Con trỏ và xâu ký tự**

Do xâu ký tự bản chất cũng là mảng các ký tự nên phần này nó cũng tương tự như mảng 1 chiều, mình sẽ nói qua 1 chút bằng 1 ví dụ đơn giản.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    int main()  {      char \*name;      name = (char \*) malloc (100\*sizeof(char));        printf("What your name? ");      gets(name);        printf("Oh, Hello %s\n", name);        return 0;  } |

**4. Con trỏ và mảng 2 chiều, mảng các con trỏ – Con trỏ đa cấp**

**a. Con trỏ và mảng 2 chiều**

Phần trên chúng ta đã tìm hiểu về con trỏ và mảng 1 chiều, và phần này con trỏ và mảng 2 chiều cũng tương tự như vậy.

Như ta đã biết thực chất trong máy tính thì bộ nhớ lưu mảng 2 chiều giống như mảng 1 chiều. Vì vậy ta hoàn toàn có thể biểu diễn mảng 2 chiều bằng con trỏ giống như mảng 1 chiều.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    int main()  {      double a[10][10], \*pa;      int n, m, i;      pa = (double \*) a;      printf("Nhap so hang va so cot:\n");      scanf("%d %d", &n, &m);        for (i = 0 ; i < n \* m; i++)      {          printf("Nhap a[%d][%d] = ", i / m, i % m);          scanf("%lf", pa + i);      }        for (i = 0 ; i < n \* m; i++)      {          if (i % m == 0) printf("\n"); // xuong dong          printf("%-5.2lf", \*(pa + i));      }        return 0;  } |

Kết quả:

Nhap so hang va so cot:

2

3

Nhap a[0][0] = 4.23

Nhap a[0][1] = 5.7

Nhap a[0][2] = 1.2

Nhap a[1][0] = 8.6

Nhap a[1][1] = 3.456

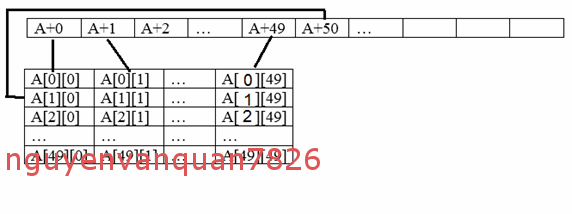
Nhap a[1][2] = 12

4.23 5.70 1.20

8.60 3.46 12.00

Ngoài ra, chúng ta có thể không cần dùng pa mà dùng ngay a là 1 con trỏ. Hoặc ta có thể nhập 1 cách tương tự như mảng 2 chiều bình thường như sau.

Các bạn lưu ý là tại sao ta lại có **pa + i\*10 + j**. Đó là vì ở hàm main() ta khai báo là a[10][10] nên máy cấp phát cho chúng ta các ô nhớ của mảng 2 chiều với 10 hàng, 10 cột mà nếu ta ko dùng hết thì nó vẫn tồn tại.



Con trỏ và mảng 2 chiều trong C

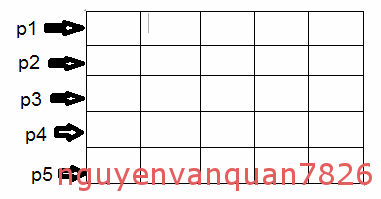
Các bạn chú ý là ta có thể viết a[i][j] nhưng **không thể viết pa[i][j]** vì theo khai báo thì a là mảng 2 chiều nên ta có thể viết như vậy còn pa là 1 con trỏ và khi ta gán pa = a thì ta đã ngầm định coi a là mảng 1 chiều. Vì vậy nếu có thì chúng ta cũng chỉ được phép viết pa[i*10+j] (lấy giá trị) và &pa[i*10+j] (lấy địa chỉ).

**b. Mảng các con trỏ**

Mảng con trỏ là một mảng chứa tập hợp các con trỏ cùng một kiểu.

Float \*a[10]; // khai báo một mảng con trỏ. Gồm 10 con trỏ: a[0], a[1], …a[9]; là 10 con trỏ.

Liên hệ với mảng 2 chiều thì ta có nhận xét sau: Mỗi hàng của mảng 2 chiều ta coi như 1 mảng 1 chiều. Mà mỗi con trỏ thì lại có mối quan hệ với 1 mảng 1 chiều, ta có hình vẽ:



Con trỏ và mảng 2 chiều trong C

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    int main()  {      double a[10][10], \*pa[10];      int n, m, i, j;        printf("Nhap so hang va so cot: ");      scanf("%d %d", &n, &m);        for (i = 0 ; i < n; i++)      {          pa[i] = a[i]; // con tro thu i tro den hang thu i          for (j = 0 ; j < m; j++)          {              printf("Nhap a[%d][%d] = ", i, j);              scanf("%lf", &pa[i][j]);          }      }        for (i = 0 ; i < n; i++)      {          printf("\n"); // xuong dong          for (j = 0 ; j < m; j++)          {              printf("%-5.2lf", pa[i][j]);          }      }        return 0;  } |

Sở dĩ ở VD này ta viết được pa[i][j] đó là do mỗi pa là 1 con trỏ đến mảng một chiều. pa[i][j] tức là phần tử thứ j của con trỏ pa[i].

Các ví dụ ở trên chúng ta đều xét khi mà khai báo mảng a[][] nên không cần cấp phát bộ nhớ cho con trỏ, bây giờ muốn cấp phát bộ nhớ cho con trỏ với mảng 2 chiều giống như cấp phát ở mảng 1 chiều thì ta làm như sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    int main()  {      double \*\*pa;      int n, m, i, j;        printf("Nhap so hang va so cot: ");      scanf("%d %d", &n, &m);        // cap phat n o nho cho n con tro (n hang)      pa = (double\*\*) malloc(n \* sizeof(double));        for (i = 0 ; i < n; i++)      {          // cap phat m o nho cho moi con tro (moi hang)          pa[i] = (double \*) malloc(m \* sizeof(double));          for (j = 0 ; j < m; j++)          {              printf("Nhap a[%d][%d] = ", i, j);              scanf("%lf", &pa[i][j]);          }      }        for (i = 0 ; i < n; i++)      {          printf("\n"); // xuong dong          for (j = 0 ; j < m; j++)          {              printf("%-5.2lf", pa[i][j]);          }      }        return 0;  } |

Đây cũng có thể coi là mảng con trỏ, hoặc con trỏ trỏ đến con trỏ (con trỏ đa cấp).

**Lập trình C: Bài 13 – Kiểu cấu trúc – struct**

**Nội dung**

* [1. Kiểu cấu trúc](https://cachhoc.net/2014/12/19/lap-trinh-c-bai-11-kieu-cau-truc-struct/#1_Kieu_cau_truc)
  + [1.1 VD mở đầu](https://cachhoc.net/2014/12/19/lap-trinh-c-bai-11-kieu-cau-truc-struct/#11_VD_mo_dau)
  + [1.2 Xây dựng kiểu cấu trúc, khai báo biến cấu trúc](https://cachhoc.net/2014/12/19/lap-trinh-c-bai-11-kieu-cau-truc-struct/#12_Xay_dung_kieu_cau_truc_khai_bao_bien_cau_truc)
  + [1.3 Truy cập đên các thành phần của cấu trúc](https://cachhoc.net/2014/12/19/lap-trinh-c-bai-11-kieu-cau-truc-struct/#13_Truy_cap_den_cac_thanh_phan_cua_cau_truc)
  + [1.4 Gán các biến có cùng kiểu cấu trúc](https://cachhoc.net/2014/12/19/lap-trinh-c-bai-11-kieu-cau-truc-struct/#14_Gan_cac_bien_co_cung_kieu_cau_truc)
* [2. Mảng cấu trúc](https://cachhoc.net/2014/12/19/lap-trinh-c-bai-11-kieu-cau-truc-struct/#2_Mang_cau_truc)
* [3. Con trỏ cấu trúc](https://cachhoc.net/2014/12/19/lap-trinh-c-bai-11-kieu-cau-truc-struct/#3_Con_tro_cau_truc)

Cách đơn giản nhất để tiếp cận về kiểu cấu trúc là xét ví dụ sinh viên. Một lớp có 100 sinh viên mỗi sinh viên gồm họ tên và mã sinh viên. Hãy nhập dữ liệu cho lớp đó. Hehe. Bạn nghĩ đến cách dùng 2 mảng: 1 mảng lưu tên, 1 mảng lưu mã sinh viên đúng không. Đúng, cách đó không sai… nhưng hãy xem yêu cầu tiếp theo… Thi học kỳ xong, hãy nhập điểm cho từng sinh viên, mỗi sinh viên gồm 10 môn (Toán, Tin, Hóa, Vật lý,…). Giờ bạn thấy sao nào… Dùng 12 mảng chăng… ồ không nên, hãy dùng **kiểu cấu trúc**. Với kiểu cấu trúc chỉ cần 1 mảng mà thôi.

**1. Kiểu cấu trúc**

Đối với mảng, chỉ có thể lưu nhiều thông tin có cùng kiểu dữ liệu. Nhưng với kiểu cấu trúc ta có thể lưu thông tin có nhiều kiểu dữ liệu khác nhau.

**1.1 VD mở đầu**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    // khai bao struct  struct sinhvien  {      char MSV[20]; // ma sinh vien      char hoten[30]; // ho ten sinh vien      double toan, tin, anh; // cac diem toan, tin, anh  };    /\* Hay thay tat ca fflush(stdin); thanh \_\_fpurge(stdin) khi ban lam tren linux\*/    int main()  {      /\* khai bao 2 bien sv1, sv2 va 1 mang       \* CNPMK10A gom 100 sinh vien       \*/      struct sinhvien sv1, sv2, CNPMK10A[100];        printf("Nhap du lieu cho sv1:\n");      printf("MSV: "); fflush(stdin);      gets(sv1.MSV);      printf("Ho ten: "); fflush(stdin);      gets(sv1.hoten);      printf("Diem toan, tin, anh: "); fflush(stdin);      scanf("%lf %lf %lf", &sv1.toan, &sv1.tin, &sv1.anh);        printf("Nhap du lieu cho sv2:\n");      printf("MSV: "); fflush(stdin);      gets(sv2.MSV);      printf("Ho ten: "); fflush(stdin);      gets(sv2.hoten);      printf("Diem toan, tin, anh: "); fflush(stdin);      scanf("%lf %lf %lf", &sv2.toan, &sv2.tin, &sv2.anh);        printf("\n --------- Thong tin sinh vien -----\n");      printf("%-20s %-30s %-7s %-7s %-7s\n", "MSV", "Ho ten", "Toan", "Tin", "Anh");      printf("%-20s %-30s %-7.2lf %-7.2lf %-7.2lf\n", sv1.MSV, sv1.hoten, sv1.toan, sv1.tin, sv1.anh);      printf("%-20s %-30s %-7.2lf %-7.2lf %-7.2lf\n", sv2.MSV, sv2.hoten, sv2.toan, sv2.tin, sv2.anh);        return 0;  } |

Kết quả:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13 | Nhap du lieu cho sv1:  MSV: DTC1  Ho ten: Pham Thi Ha  Diem toan, tin, anh: 9 9 8  Nhap du lieu cho sv2:  MSV: DTC2  Ho ten: Nguyen Van Quan  Diem toan, tin, anh: 9 9 8     --------- Thong tin sinh vien -----  MSV                  Ho ten                         Toan    Tin     Anh  DTC1                 Pham Thi Ha                    9.00    9.00    8.00  DTC2                 Nguyen Van Quan                9.00    9.00    8.00 |

Ở VD mở đầu này, chúng ta có rất nhiều điểu phải bàn

**1.2 Xây dựng kiểu cấu trúc, khai báo biến cấu trúc**

Như VD trên, để xây dựng 1 kiểu cấu trúc ta thực hiện theo cú pháp:

|  |  |
| --- | --- |
| 1  2  3  4 | struct TenKieuCauTruc  {      Khai báo các thành phần của kiểu;  }; |

Sau khi có kiểu cấu trúc rồi thì cái kiểu đó nó tương tự như 1 kiểu bình thường (int, float, char,…) và ta chỉ việc khai báo biến nữa là xong. Tuy nhiên khai báo biến thì cần có thêm từ khóa struct ở trước: (Đối với C++ thì không cần).

**struct TenKieuCauTruc TenBienCauTruc;**

Ngoài ta chúng ta còn một số cách xây dựng kiểu cấu trúc và khai báo biến cấu trúc như sau:

|  |  |
| --- | --- |
| 1  2  3  4 | struct TenKieuCauTruc  {      Khai báo các thành phần của kiểu;  } danh sách các biến thuộc kiểu cấu trúc; |

Hoặc

|  |  |
| --- | --- |
| 1  2  3  4 | struct  {      Khai báo các thành phần của kiểu;  } danh sách các biến thuộc kiểu cấu trúc ; |

Với cách khai báo này, ta bắt buộc phải khai báo các biến cấu trúc ở ngay sau cấu trúc vì không có tên kiểu cấu trúc để cho ta khai báo ở các vị trí khác nữa.

Trong phần này ta cần đề cập đến 1 từ khóa quan trọng nữa, đó là **typedef**. Từ khóa này dùng để định nghĩa 1 kiểu dữ liệu mới.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | typedef struct sinhvien  {      char MSV[20]; // ma sinh vien      char hoten[30]; // ho ten sinh vien      double toan, tin, anh; // cac diem toan, tin, anh  } kieuSinhVien; |

Khi này ta có **kieuSinhVien** là 1 kiểu dữ liệu (như int, double, …) và ta có thể khai báo các biến cấu trúc thông qua nó. Trong này có vài điều các bạn cần chú ý:

* Với **sinhvien** (kiểu cấu trúc được đặt sau từ khóa struct) thì khi khai báo biến thuộc kiểu này ta vẫn phải có từ struct ở trước nó. (VD: sinhvien svA; -> Sai còn struct sinhvien svA; -> đúng), (chú ý trong C++ thì không cần).
* Với **kieuSinhVIen** thì khi khai báo biến thuộc kiểu này ta **không được có từ struct** ở trước nó. (VD: struct kieuSinhVIen svA; -> sai, kieuSinhVIen svA; -> đúng).

Ngoài ra ta còn có thể khai báo kiểu cấu trúc lồng nhau: VD như trong 1 sinh viên có ngày sinh, trong ngày sinh lại có ngày, tháng, năm sinh.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | struct ngaysinh  {      int ngay, thang, nam;  }    typedef struct sinhvien  {      char MSV[20]; // ma sinh vien      char hoten[30]; // ho ten sinh vien      double toan, tin, anh; // cac diem toan, tin, anh      struct ngaysinh ns;  } kieuSinhVien; |

Hoặc ta khai báo ngay trong cấu trúc:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10 | typedef struct sinhvien  {      char MSV[20]; // ma sinh vien      char hoten[30]; // ho ten sinh vien      double toan, tin, anh; // cac diem toan, tin, anh      struct ngaysinh      {          int ngay, thang, nam;      } ns;  } kieuSinhVien; |

**1.3 Truy cập đên các thành phần của cấu trúc**

Để truy nhập đến thành phần của cấu trúc ta sử dụng toán tử chấm (.).

**TenBienCauTruc.TenThanhPhan;**

Như VD trên ta truy cập như sau:

sv1.hoten; sv1.toan; // truy xuất tới họ tên, điểm toán

sv1.ns.ngay; sv1.ns.thang; // truy xuất tới ngày sinh và tháng sinh.

…

Khi đã truy xuất được tới các thành phần của cấu trúc thì mỗi thành phần đó là 1 biến bình thường và ta gán giá trị hoặc nhập xuất giá trị cho chúng như cách bình thường mà chúng ta vẫn làm.

Ngoài ra nếu thành phần nào đó dài dòng thì ta có thể tránh việc dài dòng này bằng cách sử dụng từ khóa define.

VD thay vì viết:

|  |  |
| --- | --- |
| 1  2 | sv1.ns.thang;  sv1.ns.nam; |

Ta viết:

|  |  |
| --- | --- |
| 1  2  3 | #define p sv1.ns  p.thang;  p.nam; |

**1.4 Gán các biến có cùng kiểu cấu trúc**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    struct sinhvien  {      char MSV[20]; // ma sinh vien      char hoten[30]; // ho ten sinh vien      double toan, tin, anh; // cac diem toan, tin, anh  };    /\* Hay thay tat ca fflush(stdin); thanh \_\_fpurge(stdin) khi ban lam tren linux\*/    int main()  {      /\* khai bao 2 bien sv1, sv2 va 1 mang       \* CNPMK10A gom 100 sinh vien       \*/      struct sinhvien sv1, sv2, CNPMK10A[100];        printf("Nhap du lieu cho sv1:\n");      printf("MSV: "); fflush(stdin);      gets(sv1.MSV);      printf("Ho ten: "); fflush(stdin);      gets(sv1.hoten);      printf("Diem toan, tin, anh: "); fflush(stdin);      scanf("%lf %lf %lf", &sv1.toan, &sv1.tin, &sv1.anh);        sv2 = sv1; // gan gia tri cua sv1 cho sv2        printf("\n --------- Thong tin sinh vien -----\n");      printf("%-20s %-30s %-7s %-7s %-7s\n", "MSV", "Ho ten", "Toan", "Tin", "Anh");      printf("%-20s %-30s %-7.2lf %-7.2lf %-7.2lf\n", sv2.MSV, sv2.hoten, sv2.toan, sv2.tin, sv2.anh);        return 0;  } |

Sau khi gán sv2 = sv1 thì mọi thông tin của sv1 có thì sv2 cũng có. Ngoài ra ta còn có thể gán giá trị khởi đầu cho cấu trúc.

struct sinhvien sv1 = {"ABC", "Nguyen Van Quan", 9, 9, 8, {4, 5, 1992}};

Khi đó ta có các dữ liệu ban đầu của sv1 là:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | MSV: ABC  hoten: Nguyen Van Quan  toan: 9  tin: 9  anh: 8  ngày sinh: 4/5/1992. |

**2. Mảng cấu trúc**

Bên trên ta đã tìm hiểu cơ bản về kiểu cấu trúc và một vài ví dụ về cấu trúc sinhvien. Bây giờ ta tìm hiểu cách biểu diễn 1 mảng 50 sinh viên của 1 lớp học có kiểu cấu trúc như trên. Ta xét VD:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    struct sinhvien  {      char MSV[20]; // ma sinh vien      char hoten[30]; // ho ten sinh vien      double diemTB; // diem trung binh      struct ngaysinh      {          int ngay, thang, nam;      } ns;  };    int main()  {      int n = 2, i;      struct sinhvien CNPMK10A[n];        for (i = 0; i < n; i++)      {          #define sv CNPMK10A[i]          printf("Nhap du lieu cho sinh vien thu %d:\n", i + 1);          printf("MSV: "); fflush(stdin)          gets(sv.MSV);          printf("Ho ten: "); fflush(stdin);          gets(sv.hoten);          printf("Diem TB: "); fflush(stdin)          scanf("%lf", &sv.diemTB);          printf("Ngay sinh: ");          scanf("%d/%d/%d", &sv.ns.ngay, &sv.ns.thang, &sv.ns.nam);      }        printf("\n --------- Thong tin sinh vien -----\n");      printf("%-20s %-30s %-7s %-10s\n", "MSV", "Ho ten", "Diem Tb", "Ngay sinh");      for (i = 0; i < n; i++)      {          #define sv CNPMK10A[i]          printf("%-20s %-30s %-7.2lf %02d/%02d/%4d\n", sv.MSV, sv.hoten, sv.diemTB, sv.ns.ngay, sv.ns.thang, sv.ns.nam);      }      return 0;  } |

Kết quả:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | Nhap du lieu cho sinh vien thu 1:  MSV: DTC1  Ho ten: Pham Thi Ha  Diem TB: 9.2  Ngay sinh: 21/01/1993  Nhap du lieu cho sinh vien thu 2:  MSV: DTC2  Ho ten: Nguyen Van Quan  Diem TB: 9.2  Ngay sinh: 31/12/1992     --------- Thong tin sinh vien -----  MSV                  Ho ten                         Diem Tb Ngay sinh  DTC1                 Pham Thi Ha                    9.20    21/01/1993  DTC2                 Nguyen Van Quan                9.20    31/12/1992 |

**3. Con trỏ cấu trúc**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | //code by nguyenvanquan7826  #include <stdio.h>  #include <stdlib.h>    struct sinhvien  {      char MSV[20]; // ma sinh vien      char hoten[30]; // ho ten sinh vien      double diemTB; // diem trung binh      struct ngaysinh      {          int ngay, thang, nam;      } ns;  };    int main()  {      int n = 2, i;      // cap phat bo nho      struct sinhvien \*CNPMK10A = (struct sinhvien\*) malloc(n \* sizeof(struct sinhvien));        for (i = 0; i < n; i++)      {          printf("Nhap du lieu cho sinh vien thu %d:\n", i + 1);          printf("MSV: "); fflush(stdin);          gets(CNPMK10A[i].MSV);          printf("Ho ten: "); fflush(stdin);          gets(CNPMK10A[i].hoten);          printf("Diem TB: "); fflush(stdin);          scanf("%lf", &(CNPMK10A+i)->diemTB);          printf("Ngay sinh: ");          scanf("%d/%d/%d", &(CNPMK10A+i)->ns.ngay, &(CNPMK10A+i)->ns.thang, &(CNPMK10A+i)->ns.nam);      }        printf("\n --------- Thong tin sinh vien -----\n");      printf("%-20s %-30s %-7s %-10s\n", "MSV", "Ho ten", "Diem Tb", "Ngay sinh");      for (i = 0; i < n; i++)      {          #define ns CNPMK10A[i].ns          printf("%-20s %-30s %-7.2lf %02d/%02d/%4d\n", CNPMK10A[i].MSV, (\*(CNPMK10A+i)).hoten, (CNPMK10A+i)->diemTB, ns.ngay, ns.thang, ns.nam);      }      return 0;  } |

**Truy cập các thành phần cấu trúc**

Để truy cập để lấy dữ liệu các thành phần của con trỏ cấu trúc ta có 3 cách sau:

* Cách 1: CNPMK10A[i].diemTB;
* Cách 2: (\*(CNPMK10A+i)).diemTB;
* Cách 3: (CNPMK10A+i) ->diemTB;

Cả 3 cách trên đều truy cập tới DTB.

Để lấy địa chỉ ta cũng có 2 cách:

* Cách 1: &CNPMK10A[i].DTB;
* Cách 2: &(CNPMK10A+i)->diemTB

## Lập trình C: Bài 14 – Nhập xuất file trong C/C++

Trong quá trình lập trình, chúng ta thường phải đọc và ghi dữ liệu ra file. Bài này sẽ hướng dẫn các bạn cách làm đơn giản với C và C++.

Lưu ý: Trong các ví dụ dưới đây, các file input và output đặt cùng thư mục với file mã nguồn.

Ví dụ: Cho file input.txt, dòng đầu tiên gồm 1 số n là số lượng bạn bè, n dòng tiếp theo mỗi dòng là tên của 1 bạn bè. Đọc và ghi ra file output.txt danh sách bạn bè kèm số thứ tự

| **input.txt** | **output.txt** |
| --- | --- |
| 3  Nguyen Van Quan  Nguyen Thi Hong Anh  Nguyen Van Hung | 1.Nguyen Van Quan  2.Nguyen Thi Hong Anh  3.Nguyen Van Hung |

## Code C

|  |  |
| --- | --- |
| file name: test.c - code by: nguyenvanquan7826 | |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <stdio.h>    int main()  {      int n, i;      char name[255];                         // khai bao bien ten      FILE \*fi = fopen("input.txt", "r");     // mo file de doc      FILE \*fo = fopen("output.txt", "w");    // mo file de ghi        fscanf(fi, "%d", &n);                   // doc so n tu file fi        fgets(name, 255, fi);                   // loai bo dau xuong dong sau khi doc so n        for(i = 0; i < n; i++)      {          fgets(name, 255, fi);               // doc chuoi gom 255 ky tu tu file fi          fprintf(fo, "%d.%s", (i+1), name);  // ghi chuoi ra file fo      }        fclose(fi);     // dong file fi      fclose(fo);     // dong file fo        return 0;  } |

## Code C++

|  |  |
| --- | --- |
| file name: test.cpp - code by: nguyenvanquan7826 | |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <fstream>  using namespace std;    int main()  {      int n;      string name;      ifstream fi("input.txt");   // mo file de doc      ofstream fo("output.txt");  // mo file de ghi        fi >> n;                  // doc 1 so tu file      getline(fi, name);          // loai bo dau xuong dong sau khi doc so n        for (int i = 0; i <n; i++)      {          getline(fi, name);      // doc 1 dong tu file          fo << (i+1) << "." << name << '\n';     // ghi 1 dong ra file      }        fi.close();      fo.close();        return 0;  } |