Bài 5. Hàm, biến và bộ nhớ

Mục tiêu:

1. *Luyện tập cú pháp viết hàm*
2. *Thử nghiệm địa chỉ bộ nhớ của các loại biến địa phương, stack của các lời gọi hàm*
3. *Luyện tập đệ quy đơn giản.*

Giới hạn: *không dùng các thư viện stl (chẳng hạn vector, algorithm)*

A. Thực hành

Các bài này cần quan sát địa chỉ của các biến trong chương trình. Bạn sẽ cần in địa chỉ của biến ra màn hình để có thể nhìn thấy. Toán tử & cho ta địa chỉ của biến, chẳng hạn biểu thức &x có giá trị là địa chỉ của biến x khi chương trình chạy.

1. ***Địa chỉ các biến.*** Hãy viết một chương trình có khai báo một mảng a kiểu int, mảng b kiểu char, in ra màn hình địa chỉ của 03 phần tử liên tiếp của mảng a, 03 phần tử liên tiếp của mảng b. Bạn có nhận xét gì về các kết quả đó? Viết câu trả lời vào chương trình ở dạng comment.
   Khai báo thêm một vài biến trước và sau các mảng trên. Hãy thử nghiệm để trả lời câu hỏi: các biến đó có vị trí tương đối với nhau như thế nào. Viết câu trả lời vào chương trình ở dạng comment.
   Để in địa chỉ của một phần từ mảng char, bạn dùng lệnh sau, chẳng hạn cho b[i]:
   cout << (void *)&b[i];
   Nếu chỉ dùng &b[i] thì cout sẽ nghĩ rằng bạn in ra một xâu kí tự nên bạn sẽ không thấy được địa chỉ của phần từ mảng b[i]. Ta sẽ nói thêm về việc này khi học con trỏ.

2. ***Function Call Stack***. Hãy viết một chương trình dùng hàm đệ quy factorial(x) để tính giai thừa của một số nguyên N, có thể dùng thẳng code mẫu trong bài giảng. Có thể hardcode N (gán thẳng gía trị của N trong chương trình thay vì nhập từ ngoài). Trong hàm factorial, hãy in giá trị kèm theo địa chỉ của tham số x ra màn hình (có thể dùng lệnh cin << "x = " << x << " at " << &x << endl;).
   Bạn có nhận xét gì về chuỗi giá trị được in ra màn hình? Liên hệ với nội dung đã học về function call stack và vị trí của các biến địa phương trong bộ nhớ. Dựa vào kết quả, hãy thử tính xem kích thước của một stack frame cho hàm factorial là bao nhiêu. Hãy viết câu trả lời ở dạng comment trong code.

3. ***Tham trị, tham biến và đối số***. Viết một chương trình sử dụng hàm với các cơ chế truyền tham số bằng giá trị (pass-by-value) và bằng tham chiếu. Hãy dùng các lệnh in địa chỉ của biến để chứng tỏ rằng **đối số** và **tham trị** là hai biến độc lập có địa chỉ khác nhau, trong khi **đối số** và **tham biến** chính là một biến. Hãy viết câu trả lời ở dạng comment trong code.

4. ***Tham số là mảng, string***. Dùng chương trình thử nghiệm để trả lời các câu hỏi:

a. mảng được truyền theo cơ chế nào pass-by-value hay pass-by-reference?.
b. C++ string được truyền theo cơ chế nào pass-by-value hay pass-by-reference?.

Hãy viết câu trả lời ở dạng comment trong code.

5. ***Biến tham chiếu***. Hãy viết một chương trình dùng thử nghiệm để trả lời các câu hỏi sau đây: (viết câu trả lời ở dạng comment cạnh đoạn code thử nghiệm tương ứng)
    a. .Biến tham chiếu và biến và nó chiếu tới là một biến hay hai biến trong bộ nhớ?
    b. Có thể khai báo một tham chiếu mà chưa chiếu ngay nó tới một biến thường hay không?
    c. Có thể chiếu một tham chiếu tới một biến khác với đích ban đầu của nó hay không? (ví dụ lúc đầu b chiếu tới a, sau chiếu b tới c)
6. ***Tìm kiếm nhị phân***. Hãy cài thuật toán tìm kiếm nhị phân theo hai cách (dùng vòng lặp và dùng đệ quy - xem chi tiết thuật toán tại bài giảng) trong hai hàm, mỗi hàm nhận 4 tham số: key cần tìm, mảng cần tìm, hai chỉ số low, high của đoạn cần tìm, viết hàm main thử nghiệm hai hàm trên. Để tiện, không nhập dữ liệu mà hãy hardcode mảng số nguyên và các giá trị key cần tìm (khai báo và khởi tạo sẵn mảng a chứa các số nguyên đã sắp xếp tăng dần). Dùng code hàm main để test các trường hợp key không có trong mảng, key nằm đầu mảng, key nằm cuối mảng, key nằm chính giữa mảng, và key nằm ở vị trí ngẫu nhiên.
Hãy đặt đồng hồ đo thời gian kể từ lúc bạn bắt đầu code đến khi pass tất cả các test là bao nhiêu phút., ghi vào README.txt

B. Câu hỏi

1. Hãy viết một hàm majority() lấy ba tham số kiểu bool, và trả về true nếu có ít nhất hai tham số có giá trị true, nếu không thì trả về false. Không được dùng lệnh if trong hàm.

2. Hãy viết một hàm eq() lấy ba tham số: hai mảng int một chiều và một số nguyên n. Hàm trả về true nếu n phần tử của hai mảng bằng nhau đôi một, nếu không thì trả về false.

3. Cho hàm cube sau:
```
void cube(int i) {
   i = i * i * i;
}
```
Hỏi trong vòng lặp sau, hàm cube() chạy bao nhiêu lần?
```
for (int i = 0; i < 1000; i++)
   cube(i);
```

4. Chương trình sau đây cho kết quả gì?
```
void negate(int a) {
    a = -a;
}

int main() {
```

```
    int a = 17;
    cout << a;
    negate(a);
    cout << a;
}
```

5. Hãy lập một bộ test để thử xem trong các hàm dưới đây, đâu là cài đặt đúng cho một hàm min trả về số nhỏ nhất trong 04 tham số. Có thể hardcode giá trị các test thẳng vào chương trình test.

```
int min(int a, int b, int c, int d) {
   // if a is the smallest return it
   if (a <= b && a <= c && a <= d) return a;

   // otherwise, if b is the smallest of b, c, and d, return it
   if (b <= c && b <= d) return b;

   // otherwise, return the smaller of c and d
   if (c <= d) return c;
   return d;
}

int min(int a, int b, int c, int d) {
   int min = a;
   if (b < min) min = b;
   if (c < min) min = c;
   if (d < min) min = d;
   return min;
}

int min(int a, int b, int c, int d) {
   if (a < b && a < c && a < d) return a;
   if (b < c && b < d)          return b;
   if (c < d)                   return c;
   return d;
}

int min(int a, int b, int c, int d) {
   if (a <= b) {
      if (a <= c) {
         if (a <= d)  return a;
         else return d;
      }
      if (c <= d) return c;
      else return d;
   }
   if (b <= c) {
      if (b <= d) return b;
      else return d;
```

```
    }
    else if (c <= d) return c;
    return d;
}

int min(int a, int b) {
    if (a <= b) return a;
    else        return b;
}
int min(int a, int b, int c, int d) {
    return min(min(a, b), min(c, d));
}
```

6. Cho hai hàm overload sau:

```
void f(int x, double y) {
    cout << "f(int, double)";
}

void f(double x, int y) {
    cout << "f(double, int)";
}
```

Lời gọi hàm này sẽ cho kết quả gì?

```
f(1,2);
```

7. Tác dụng của lệnh return là gì?
   A. Dùng khi muốn thoát khỏi hàm
   B. Trả lại kết quả cho hàm
   C. Bắt buộc phải có trong hàm
   D. Cả A và B

8. Có thể truyền vào bao nhiêu đối số trong một hàm?
   A. Bao nhiêu cũng được.
   B. Ít nhất là 1.
   C. Nhiều nhất là 10.
   D. Chỉ duy nhất 1.

9. Khai báo prototype nào sau đây là không hợp lệ?
   A. int funct(char x, char y);
   B. double funct(char x)
   C. void funct();
   D. char x();

10. Kiểu trả về của hàm với prototype: "int func(char x, float v, double t);" là:
    A. char
    B. int

C. float
D. double

11. Lời gọi hàm nào sau đây là hợp lệ (giả sử hàm đấy tồn tại)?
    A. funct;
    B. funct x, y;
    C. funct();
    D. int funct();

12. Đâu là định nghĩa một hàm hoàn chỉnh?
    A. int funct();
    B. int funct(int x) {return x=x+1;}
    C. void funct(int) {cout<<"Hello"}
    D. void funct(x) {cout<<"Hello"}

13. Giả sử một hàm tên là function1 có một biến tên là sam được khai báo bên trong định nghĩa hàm function1, và một hàm tên là function2 cũng có một biến tên là sam được khai báo bên trong định nghĩa của function2. Điều gì sẽ xảy ra khi chạy chương trình (giả thiết mọi thứ khác đều ổn):
    A. Không biên dịch được;
    B. Biên dịch được nhưng không chạy được;
    C. Chạy được nhưng sẽ gặp lỗi logic không mong muốn;
    D. Chạy đúng như mong muốn;

14. Cho biết kết quả của đoạn code sau:

```
#include <iostream>
void display (int k)
{
  int j;
  std::cout << “Hello” << std::endl;
}
int main(int argc, char** argv)
{
  int s;
  display(s);

  return 0;
}
```

    A. Chương trình gặp lỗi biên dịch vì 2 biến s, k không giống nhau
    B. Chương trình gặp lỗi biên dịch vì 2 biến s, k không được khởi tạo trước khi truyền cho hàm
    C. Chương trình biên dịch thành công nhưng gặp lỗi khi chạy vì 2 biến s, k không giống nhau
    D. Chương trình biên dịch thành công và kết quả khi chạy là : Hello

15. Đoạn code sau in ra kết quả như thế nào?

```
#include <iostream>
int foo(int y);
int foo(int x)
{
  return x+1;
}
int main(int argc, char** argv)
{
  int x = 3;
  int y = 6;

  std::cout << foo(x) << std::endl;

  return 0;
}
```

A. 3
B. 4
C. 9
D. Lỗi

16. Hàm inline được định nghĩa trong
    A. Thời gian chạy
    B. Thời gian biên dịch
    C. Tùy vào nó được gọi khi nào
    D. Cả B và C

17. Nếu một hàm được viết để sử dụng trong một chương trình sẵn có thì nó có thể được đưa vào thư viện hàm để các chương trình khác có thể sử dụng được
    A. Đúng
    B. Sai

18. Số tham số và thứ tự các tham số thật phải đúng như các tham số hình thức trong khai báo hàm
    A. Đúng
    B. Sai

19. Không cần chỉ định kiểu trả về của hàm khi hàm đó trả về kiểu dữ liệu nào sau đây:
    A. int
    B. void
    C. float
    D. char

20. Định nghĩa hàm nào sau đây không hợp lệ
    A. void foo(){}
    B. void foo(void){}
    C. foo(void){}
    D. void foo{}

21. Khi gọi hàm, các tham số được truyền vào phải giống các tham số hình thức ở
    A. Kiểu trả về
    B. Kiểu dữ liệu
    C. Tên
    D. Giá trị

22. Nếu một hàm được gọi trước khi nó được định nghĩa thì điều kiện là gì?
    A. Kiểu trả về của hàm phải là kiểu void
    B. Kiểu đầu vào của hàm phải là kiểu void
    C. Trước khi gọi hàm nó phải được khai báo
    D. Không thể được.

23. Hàm main() là một ví dụ về
    A. Hàm sơ cấp
    B. Hàm kiểu void
    C. Hàm do người dùng định nghĩa
    D. Hàm thư viện (pre-defined)

24. Một hàm không trả về giá trị thì không thể có mặt trong biểu thức
    A. Đúng
    B. Sai

25. Phát biểu nào sau đây là không đúng?
    A. Hàm có thể được định nghĩa trong hàm main()
    B. Hàm có thể được khai báo trong hàm main()
    C. Hàm có thể được gọi trước khi nó được định nghĩa
    D. Hàm chỉ có thể được định nghĩa bên ngoài hàm main()

26. Phát biểu nào sau đây về hàm main() là đúng
    A. Là hàm được khai báo đầu tiên khi thực hiện chương trình
    B. Là hàm luôn được gọi đầu tiên khi chương trình thực hiện
    C. Là hàm không được phép truyền tham số vào
    D. Là hàm có thể gọi bình thường như các hàm người dùng định nghĩa khác.

C. Bài tập

Tất cả các bài trong phần này đều cần dùng hàm một cách thích hợp, không hàm nào dài quá 15 dòng.

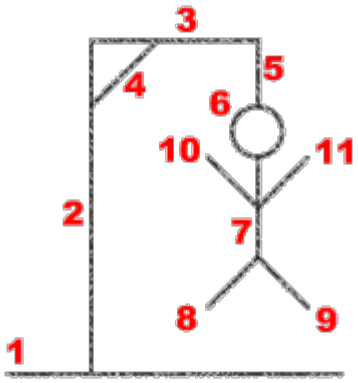
1. ***Dò mìn đơn giản***. Viết chương trình trò chơi dò mình. Input từ bàn phím gồm một cặp số m, n, K nguyên dương nhỏ hơn 10. Chương trình sinh ngẫu nhiên một bản đồ mìn kích thước m hàng n cột, trên đó có K quả mìn. Lặp đi lặp lại cho đến hết ván: Người chơi nhập một tọa độ x, y. Với mỗi tọa độ đọc được, chương trình kiểm tra xem tọa độ đó có mìn hay không. Nếu có thì báo 'YOU'RE DEAD!', in lại bản đồ với tất cả các quả mìn ra màn hình và kết thúc trò chơi. Nếu không có thì coi như người chơi đã mở thành công ô đó, chương trình in ra bản đồ với các ô đã thử mà không có mìn, ghi tại mỗi ô đó số mìn nằm trong 8 ô xung quanh. ….. Đại loại như trò chơi Dò mìn thông thường nhưng không cần loang khi thử vào ô không có mìn tại đó và xung quanh.

2. ***Dò mìn chuẩn (**):*** Tương tự như trên nhưng cần loang ra xung quanh như trò Dò mìn thông thường.

3. ***Banner.*** Viết lại chương trình cho bài C7 (ArraysAndString), lần này dùng hàm, chẳng hạn
   a. hàm draw(image, width, height, x, y) vẽ hình ảnh trong mảng image[width][height] và tọa độ x,y;
      Ví dụ sử dụng: dùng để vẽ hình chữ a ở dạng mảng vào tọa độ 1,1.
   b. hàm draw(text, x,y) vẽ xâu kí tự text vào tọa độ x,y. Hàm này sẽ gọi hàm draw(image….) ở trên để vẽ từng kí tự trong chuỗi text vào tọa độ thích hợp.
      Ví dụ sử dụng: gọi draw("kevin", 0,0) để vẽ đoạn text "kevin" vào góc trái trên.

```
#    #  ######  #    #      #    #    #
#   #   #       #    #      #    ##   #
####     #####   #    #      #    # #  #
#  #     #       #    #      #    #  # #
#   #    #       #  #  #     #    #   ##
#    #   ######    ##        #    #    #
```

4. ***Hangman***. Viết chương trình nhập input là một chuỗi các từ tiếng Anh từ input chuẩn,input kết thúc bằng một dấu chấm '.', tất cả cách nhau bởi dấu trắng (space, newline, tab... (nên điều hướng từ file), sau đó chương trình sẽ chọn ngẫu nhiên một từ và cho người dùng chơi trò hangman. Mỗi lần cho người dùng nhập một kí tự (không phân biệt chữ hoa, chữ thường) và thông báo kết quả (trạng thái từ đang đoán và số bước đi còn lại). Trò chơi kết thúc khi người chơi đã đi hết 10 bước hoặc khi người chơi đã đoán xong (tùy xem cái gì đến trước). Tùy bạn tự thiết kế cách hiển thị ra màn hình, vẽ hình bằng text cũng được, mà chỉ hiện số bước còn lại chứ không vẽ cũng được.



5. Viết hàm kiểm tra xem 1 số nguyên dương có phải số nguyên tố hay không. Hàm nhận 01 tham số là số cần kiểm tra. Hàm trả về giá trị 0 nếu đó không là số nguyên tố; giá trị 1 nếu đó là số nguyên tố.
   Sử dụng hàm vừa định nghĩa để in ra tất cả các số nguyên tố nhỏ hơn N (với N nhập vào từ bàn phím).
6. Viết hàm tính ước chung lớn nhất của hai số nguyên dương. Hàm nhận vào 02 tham số là hai số nguyên dương cần tính ước chung lớn nhất. Hàm trả về ước chung lớn nhất của 2 số này.
   Sử dụng hàm vừa định nghĩa để kiểm tra 2 số a & b có nguyên tố cùng nhau hay không (với a, b là 2 số nguyên dương được nhập vào từ bàn phím).
7. Viết hàm in ra 1 dòng có M dấu cách (“ ”) sau đấy đến N dấu * (“*”) với M và N là 2 tham số.
   Sử dụng hàm vừa định để in ra tam giác hoa thị như sau:

```
    *
   ***
  *****
 *******
*********
```

   Số dòng cần in được nhập vào từ bàn phím. Như ví dụ trên số dòng cần in là 5.

8. Viết hàm làm tròn (rnd) một số thực (double) về một số nguyên (int) gần nhất. Hàm nhận vào một số thực và trả về một số nguyên gần nó nhất.
   Yêu cầu viết 02 định nghĩa khác nhau cho hàm này:
   1. Sử dụng hàm ceil / floor (có trong thư viện <math.h>)
   2. Không sử dụng ceil / floor

9. Hàm rand() trong thư viện <stdlib.h> trả về một số nguyên ngẫu nhiên. Sử dụng hàm này để viết một hàm trả về một số ngẫu nhiên nhỏ hơn N (với N là tham số được truyền vào hàm).

10. Sử dụng hàm rand() (trả về một số nguyên ngẫu nhiên) trong thư viện <math.h> để tạo một mảng N số nguyên trong khoảng từ 0 – 49. Viết hàm in ra tất cả các bộ 3 số nguyên (trong mảng vừa tạo) có tổng chia hết cho 25.

11. Viết hàm đổi số ở hệ cơ số 10 sang hệ cơ số 2 và ngược lại. Tham khảo phương pháp đổi tại http://vi.wikipedia.org/wiki/Hệ_nhị_phân.

12. Viết chương trình tính tổ hợp, thực hiện nhưng công việc sau
Viết hàm tính tổ hợp chập k của n phần tử: int toHop(int k, int n);.

Viết hàm nhập dữ liệu cho k & n thỏa mãn điều kiện 0 ≤k ≤n, 1≤n ≤20:
// trả về 1 nếu dữ liệu thỏa mãn điều kiện, nếu không thỏa mãn trả về 0
int kiemTra(int k, int n);

Viết hàm nhập vào các cặp số k & n gồm nhiều dòng, mỗi dòng chứa 2 số nguyên dương k & n là 2 số cần tính tổ hợp. Kết thúc việc nhập dữ liệu bằng 2 số -1 -1. Gợi ý: dùng 2 mảng 1 chiều để lưu k & n.
void nhapKN(int k[], int n[], int *soPhanTu);

Sau đó in ra tổ hợp chập k của n phần tử cho từng cặp k & n vừa nhập.
Dữ liệu vào:
2 10
10 10
-1 -1
Kết quả ra màn hình:
45
1