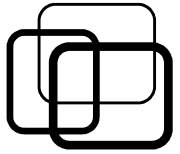
Hàm và cách tổ chức chương trình C

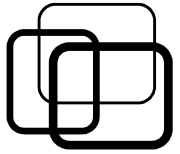
GV. Nguyễn Minh Huy

Nội dung



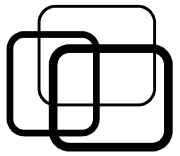
- Khái niệm hàm.
- Truyền tham số và tầm vực.
- Tổ chức chương trình C.

Nội dung



- **Khái niệm hàm.**
- Truyền tham số và tầm vực.
- Tổ chức chương trình C.

Khái niệm hàm

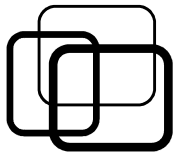


- Xét chương trình sau:
  - Nhập vào 3 số nguyên a, b, c >= 0.
  - Tính và xuất S = a! + b! + c!.

  ➔ Hãy chỉ ra những phần trùng lắp của chương trình.
- Điểm yếu của chương trình trùng lắp:
  - Tốn thời gian, công sức.
  - Khi có thay đổi → sửa nhiều chỗ.

  ➔ Viết 1 lần, tái sử dụng lại nhiều lần.

Khái niệm hàm



- Phương pháp đặt hàm:

- B1: S1 = 1.
- B2: Lặp **a** > 0
  S1 = S1 * **a**.
  a = **a** – 1.

- B3: S2 = 1.
- B4: Lặp **b** > 0
  S2 = S2 * **b**.
  b = **b** – 1.

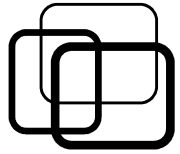
- B5: S3 = 1.
- B6: Lặp **c** > 0
  S3 = S3 * **c**.
  c = **c** – 1.

- B7: **S = S1 + S2 + S3**.

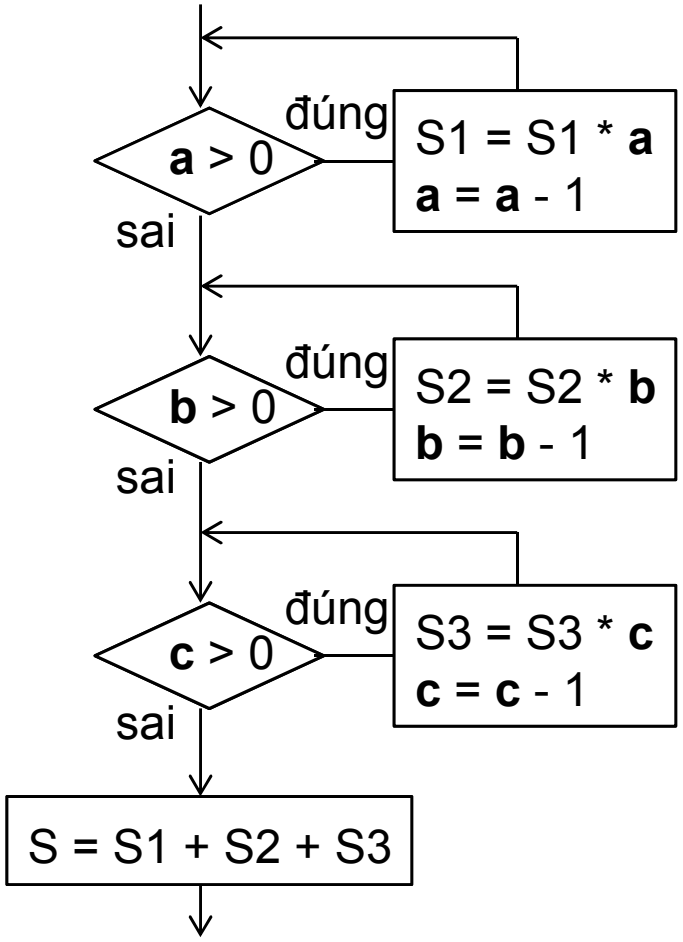
- B0: Đặt **F(x)** như sau:
  - B0.1: T = 1.
  - B0.2: Lặp **x** > 0
    T = T * **x**.
    x = **x** – 1.

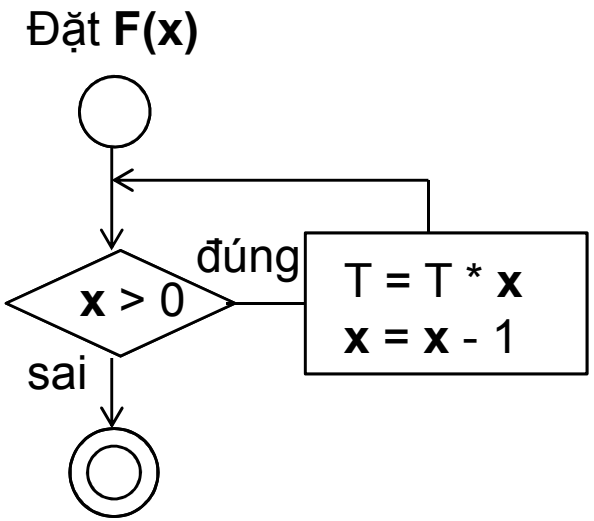
- B1: S1 = **F(a)**.
- B2: S2 = **F(b)**.
- B3: S3 = **F(c)**.
- B4: S = S1 + S2 + S3.

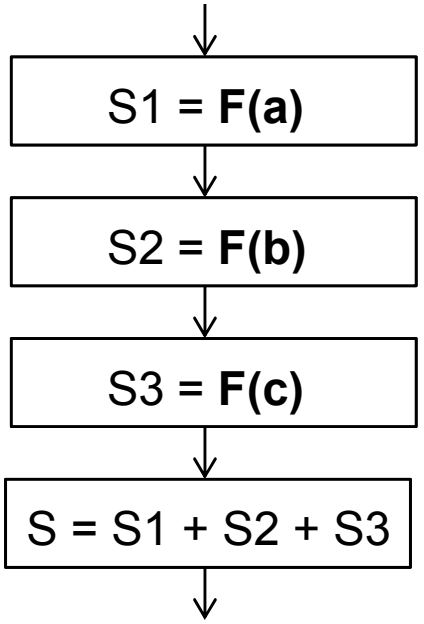
Khái niệm hàm



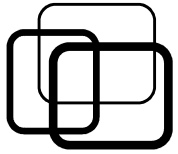
- Phương pháp đặt hàm:



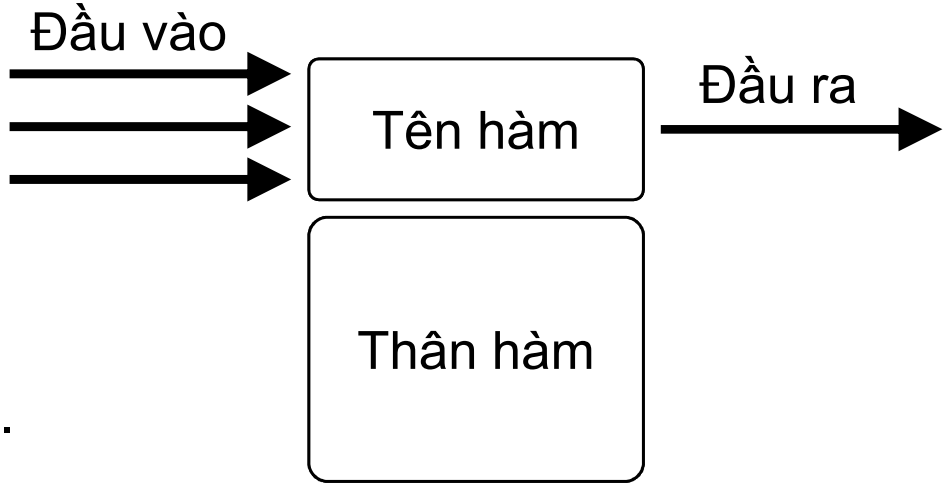




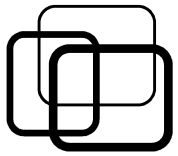
Khái niệm hàm



- Hàm trong ngôn ngữ C:
  - Một khối lệnh được đặt tên.
  - Có thể gọi từ bất kỳ đâu trong chương trình.
  - Có thể gọi nhiều lần với tham số khác nhau.
  - Cấu trúc của hàm:
    - Phần khai báo:
      - Tên hàm.
      - Tham số đầu vào.
      - Kết quả đầu ra.
      - ➔ Định danh hàm.
    - Phần cài đặt: thân hàm.



Khái niệm hàm



- Các bước sử dụng hàm trong C:
  - Khai báo hàm (prototype):

    <*Kiểu trả về*> **<Tên hàm>(** <*Khai báo tham số*> **)**;

    <Kiểu trả về>: int, float, char, …, **void** (không trả về).

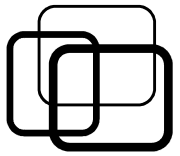
    float **tinhDTB(** float van, float toan **)**;
    void **xuatKetQua**( );
  - Cài đặt hàm:

    <*Kiểu trả về*> **<Tên hàm>(** <*Khai báo tham số*> **)**
    {
    [Các câu lệnh]
    [**return** <*giá trị trả về*>;]
    }
  - Gọi thực hiện hàm:

    <Tên hàm>(<*Các tham số*> **)**;

    float dtb = **tinhDTB**(7, 8.5);

Khái niệm hàm

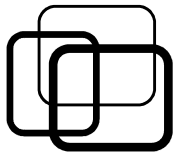


- Các bước sử dụng hàm trong C:

```
#include <stdio.h>

// Khai báo hàm.
long tinhGT(int n);

int main()
{
        /* Viết khai báo biến.
           Viết lệnh nhập a, b, c. */

        // Gọi thực hiện hàm.
        S1 = tinhGT(a);
        S2 = tinhGT(b);
        S3 = tinhGT(c);

        S = S1 + S2 + S3;
}
```

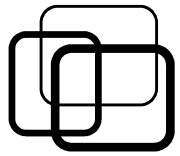
```
// Cài đặt hàm.
long tinhGT(int n)
{
        long s = 1;
        for ( ; n > 0; n--)
                s = s * n;
        return s;
}
```

Nội dung



- Khái niệm hàm.
- **Truyền tham số và tầm vực.**
- Tổ chức chương trình C.

Truyền tham số và tầm vực

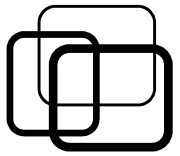


- Cách truyền tham số vào hàm:
  - Truyền tham trị (pass-by-value):
    - Truyền giá trị vào hàm.
    - Hàm chỉ nhận bản sao của tham số.
    - Tham số KHÔNG THAY ĐỔI sau khi truyền.
    - Tham số: biến, hằng, biểu thức.

```
float tinhDTB( float van, float toan )
{
        van = van * 2;
        toan = toan * 3;
        return (van + toan) / 5;
}
```

```
int main()
{
        float van, toan, dtb;

        dtb = tinhDTB( van, toan );
        dtb = tinhDTB( 6, 8.5 );
        dtb = tinhDTB( van + 1, toan );
        // Biến van, toan không thay đổi.
}
```

Truyền tham số và tầm vực

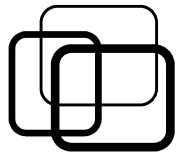


- Cách truyền tham số vào hàm:
  - Truyền tham chiếu (pass-by-reference) (C++):
    - Còn gọi là truyền tham biến.
    - Hàm nhận bản gốc của tham số.
    - Tham số CÓ THỂ THAY ĐỔI sau khi truyền.
    - Tham số chỉ có thể là biến.
    - Khai báo tham số: **&**<tên tham số>.

```
float tinhDTB( float &van, float toan )
{
        van = van * 2;
        toan = toan * 3;
        return (van + toan) / 5;
}
```

```
int main()
{
        float van, toan, dtb;

        dtb = tinhDTB( van, toan );
        // Biến van bị thay đổi.
        dtb = tinhDTB( 6, 8.5 );          //Sai
        dtb = tinhDTB(van + 1, toan); //Sai
}
```

Truyền tham số và tầm vực



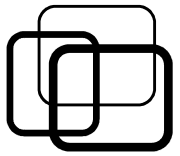
- Cách truyền tham số vào hàm:
  - Ghi chú:
    - Dùng truyền tham chiếu để trả về giá trị.

      ➔ Hàm có nhiều giá trị trả về.

```
void nhapDiem( float &diem1, float &diem2 )
{
        printf("Nhap diem van = ");
        scanf("%d", &diem1);
        printf("Nhap diem toan = ");
        scanf("%d", &diem2);
}
void tinhDTB( float van, float toan, float &dtb )
{
        van = van * 2;
        toan = toan * 3;
        dtb = (van + toan) / 5;
}
```

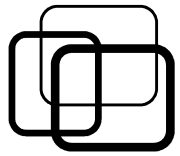
```
int main()
{
        int   van, toan;
        float dtb;

        // van, toan thay đổi
        nhapDiem(van, toan);

        // dtb thay đổi
        tinhDTB(van, toan, dtb);
}
```

Truyền tham số và tầm vực



- Tầm vực:
  - Phạm vi hoạt động của biến và hàm.
  - Phân loại:
    - Toàn cục: hoạt động trên toàn chương trình.
    - Cục bộ: hoạt động trong một khối lệnh.
  - Hàm → phạm vi toàn cục.
  - Biến:
    - Biến toàn cục: khai báo ngoài hàm (kể cả hàm main).
      ➔ Hoạt động trên toàn chương trình.
    - Biến cục bộ: khai báo trong thân hàm hoặc khối lệnh.
      ➔ Hoạt động trong thân hàm hoặc khối lệnh.

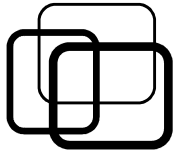
Truyền tham số và tầm vực



- Tầm vực:

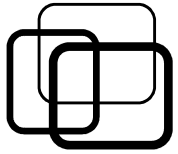
```
float  S;                          // Khai báo biến toàn cục.
int tinh();                        // Khai báo hàm.

void main()
{
      int  a = S + tinh();         // Biến cục bộ hàm main.
      while (a > 0)
      {
            int  b = S + tinh();   // Biến cục bộ vòng lặp.
      }
}

int hamXYZ()
{
      int  y = S * 2;              // Biến cục bộ hàm tinh.
}
```

Nội dung



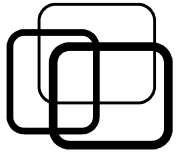
- Khái niệm hàm.
- Truyền tham số và tầm vực.
- **Tổ chức chương trình C.**

Tổ chức chương trình C



- Một quyển sách được tổ chức thế nào?
  - Không thể viết tất cả trên một trang giấy!!
    - ➔ Chia làm nhiều chương.
    - ➔ Có tóm tắt ở đầu.
    - ➔ Nội dung các chương ở sau.

Tổ chức chương trình C



- Cách tổ chức chương trình C:
  - Tổ chức giống một quyển sách.
    - Các chương ~ các file mã nguồn.
    - Tóm tắt ~ hàm main.

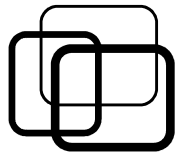
  ➔ Làm sao kết nối các file mã nguồn?

```
// File main.cpp
void main()
{
        nhap();
        tinhToan1();
        tinhToan2();
        xuat();
}
```

```
// File nhapxuat.cpp
void nhap()
{
}

void xuat()
{
}
```

```
// File xuly.cpp
int tinhToan1()
{
}

int tinhToan2()
{
}
```

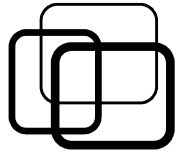
Tổ chức chương trình C



- Header file:
  - Kết nối các file mã nguồn trong chương trình.
  - Làm mã nguồn trên các file hiểu lẫn nhau.
  - Có đuôi file **.h**.
  - Cách sử dụng:
    - Tạo file **.h** cho file mã nguồn **.cpp**.
    - File **.h** chỉ chứa **khai báo** hàm và biến toàn cục.
    - File **.cpp** chỉ chứa **cài đặt** hàm.
    - Để A.cpp hiểu mã nguồn của B.cpp

      ➔ Trong A.cpp dùng **#include** “<Đường dẫn B.h>”

Tổ chức chương trình C



- Header file:

```
// File main.cpp
#include "nhapxuat.h"
#include "xuly.h"

int main()
{
        nhap();
        tinhToan1();
        tinhToan2();
        xuat();
}
```

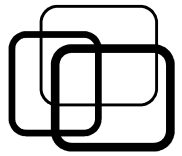
```
// File nhapxuat.h
// Khai báo hàm
void nhap();
void xuat();
```

```
// File xuly.h
// Khai báo hàm
int tinhToan1();
int tinhToan2();
```

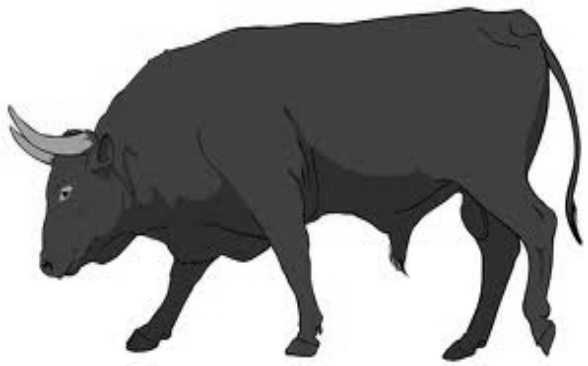
```
// File nhapxuat.cpp
#include "nhapxuat.h"
void nhap()
{
}
void xuat()
{
}
```

```
// File xuly.cpp
#include "xuly.h"
int tinhToan1()
{
}
int tinhToan2()
{
}
```

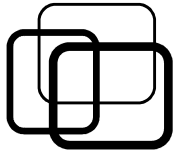
Tổ chức chương trình C



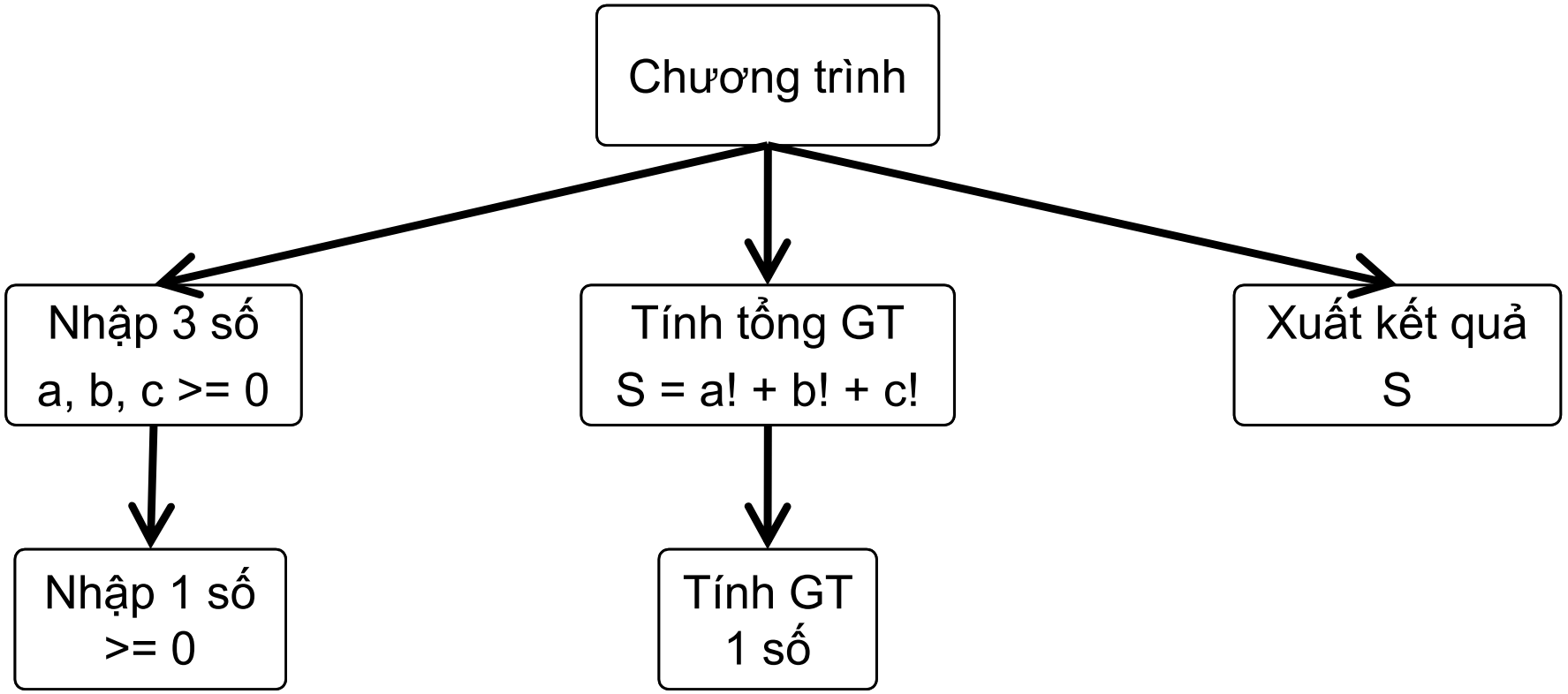
- Phương pháp "chia để trị":
  - Làm sao để ăn hết một con bò?
    - ➔ Chia thành từng phần nhỏ.
    - ➔ "Giải quyết" từng phần.
  - Phần thế nào là đủ nhỏ?
    - ➔ "Nhai, nuốt" được.
  - Để viết một chương trình:
    - Chia thành từng hàm nhỏ.
    - Xử lý từng hàm.
    - Mỗi hàm nên chỉ 10-20 câu lệnh!!



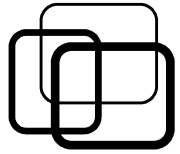
Tổ chức chương trình C



- Cây phân rã chương trình:
  - Nhập vào 3 số nguyên a, b, c >= 0.
  - Tính và xuất S = a! + b! + c!.



Tổ chức chương trình C



// File **xuly1.h**
```
void nhap3so(int a, int b, int c);
long tinhTGT(int a, int b, int c);
void xuatKQ(long ketqua);
```

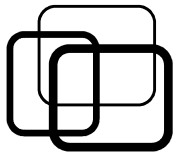
// File **xuly2.h**
```
void nhap1so(int x);
long tinhGT(int n);
```

// File **main.cpp**
```
#include “xuly1.h”

int main()
{
        int    a, b, c;
        long S;

        nhap3so(a, b, c);
        S = tinhTGT(a, b, c);
        xuatKQ(S);
}
```

// File **xuly1.cpp**
```
#include “xuly1.h”
#include “xuly2.h”
void nhap3so(int a, int b, int c)
{
        nhap1so(a);
        nhap1so(b);
        nhap1so(c);
}
long tinhTGT(int a, int b, int c)
{
        return tinhGT(a) +
            tinhGT(b) + tinhGT(c);
}
void xuat(long ketqua)
{
        printf(“S = %ld”, ketqua);
}
```

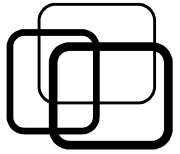
// File **xuly2.cpp**
```
#include “xuly2.h”
#include <stdio.h>
void nhap1so(int &x)
{
        do {
                printf(“Nhap 1 so = “);
                scanf(“%d”, &x);
        } while (x < 0);
}

long tinhGT(int n)
{
        long S = 1;
        for ( ; n > 0; n--)
                S = S * n;
        return S;
}
```

Tóm tắt



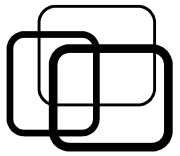
- Khái niệm hàm:
  - Một khối lệnh được đặt tên.
  - Có thể gọi từ nhiều nơi trong chương trình.
- Cách truyền tham số:
  - Cách thức đưa giá trị từ bên ngoài vào hàm.
  - Truyền tham trị:
    - Truyền giá trị vào hàm.
    - Tham số: biến, hằng, biểu thức.
  - Truyền tham chiếu:
    - Truyền biến vào hàm.
    - Tham số: biến.

Tóm tắt



- Tầm vực:
  - Phạm vi hoạt động của biến và hàm.
  - Hàm: toàn cục.
  - Biến: toàn cục hoặc cục bộ.
- Tổ chức chương trình C:
  - Header file:
    - File đuôi .h, dùng kết nối các file mã nguồn.
    - A.cpp muốn hiểu B.cpp → #include “B.h”
  - “Chia để trị”:
    - Chia chương trình thành các hàm.
    - Mỗi hàm nên tối đa 10 câu lệnh.
    - Cây phân rã chương trình.

Bài tập



Bài tập 6.1:

Viết chương trình C tìm số nguyên tố như sau:

(tổ chức theo dạng hàm và chia làm nhiều file):

- Nhập vào số nguyên dương N.
- Đếm và xuất ra tất cả các số nguyên tố <= N.

Định dạng nhập:

Nhap N = 11

Định dạng xuất:

#1 = 2

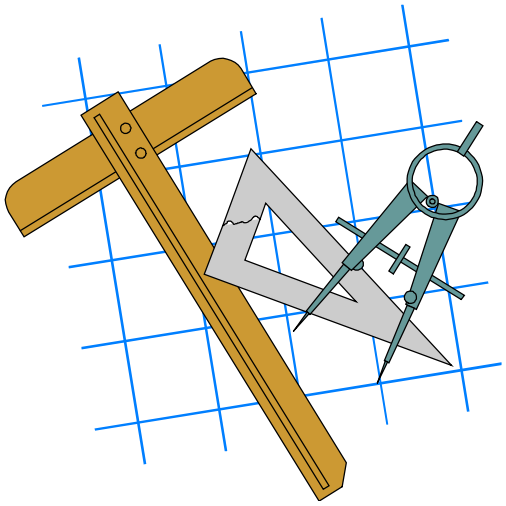
#2 = 3

#3 = 5

#4 = 7

#5 = 11

Co tat ca 5 so nguyen to.



Bài tập

Bài tập 6.2:

Viết chương trình C tính toán trên phân số như sau:

(tổ chức theo dạng hàm và chia làm nhiều file):

- Nhập vào 2 phân số a/b và c/d.
- Nhập vào phép tính (+, -, *, /).
- Thực hiện phép tính trên 2 phân số và xuất kết quả.

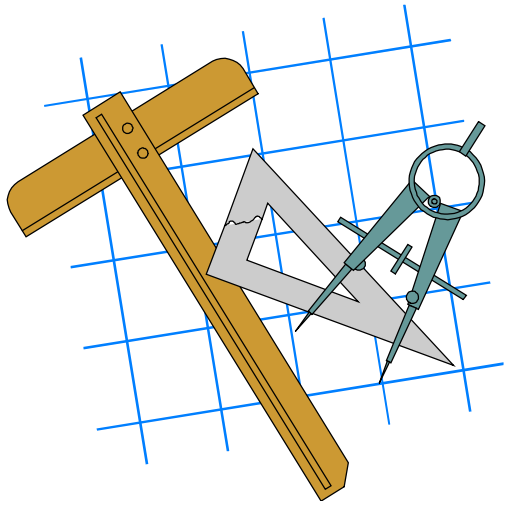
Định dạng nhập:

Phan so 1 (a/b) = 1 2

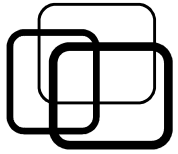
Phan so 2 (c/d) = 3 4

*Phep tinh (+, -, *, /) = +*

Ket qua = 10/8



Bài tập



Bài tập 6.3:

Viết chương trình C phân loại tam giác như sau:

(tổ chức theo dạng hàm và chia làm nhiều file):

- Nhập vào 3 số thực dương a, b, c.
- Kiểm tra xem a, b, c có lập thành một tam giác không.
- Nếu có, hãy cho biết đó là tam giác gì.

Định dạng nhập:

Nhap a, b, c = 3 4 5

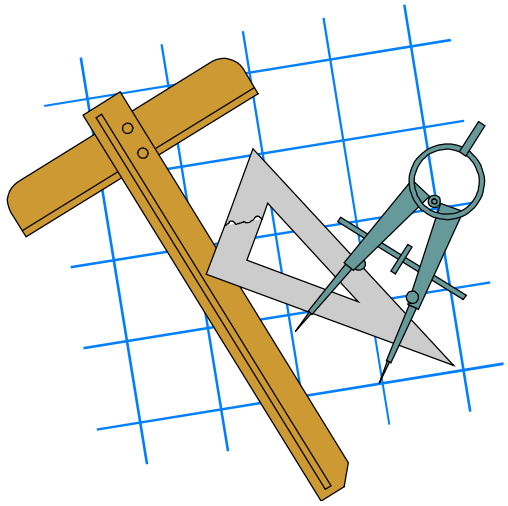
Định dạng xuất:

Lap thanh tam giac.

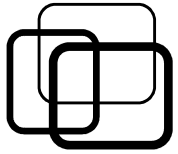
Loai tam giac la vuong.

Định dạng xuất không thỏa tam giác:

Khong lap thanh tam giac!



Bài tập



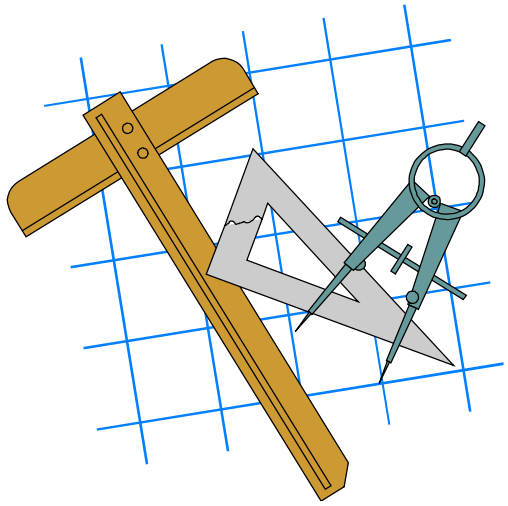
Bài tập 6.4:

Viết chương trình C tính tiền điện như sau:

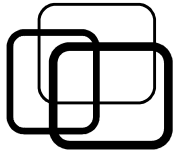
(tổ chức theo dạng hàm và chia làm nhiều file):

- Nhập vào chỉ số điện cũ và chỉ số điện mới (theo kWh).
- Tính tiền điện và xuất kết quả.

Bậc	kWh tiêu thụ	Đơn giá
1	0 – 100	1549 đ
2	101 – 150	1600 đ
3	151 – 200	1858 đ
4	201 – 300	2340 đ
5	301 – 400	2615 đ
6	401 – trở lên	2701 đ



Bài tập



Bài tập 6.5:

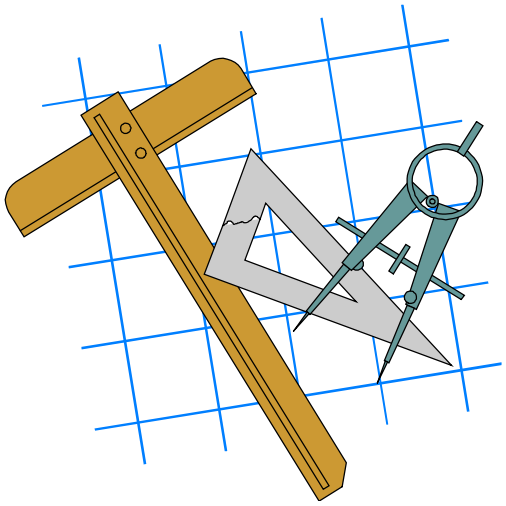
Viết chương trình C mô phỏng menu lựa chọn như sau:

(tổ chức theo dạng hàm và chia làm nhiều file):

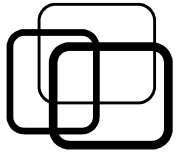
- Xuất ra menu như sau:

1. Bai tap 6.1. *2. Bai tap 6.2.* *3. Bai tap 6.3.* *4. Bai tap 6.4.* *5. Thoat.*
Lua chon cua ban (1-5):

- Nhập vào lựa chọn là số nguyên từ 1 đến 5.
- Lựa chọn từ 1 đến 4:
  - + Thực hiện bài tập tương ứng.
  - + Trở lại menu để lựa chọn tiếp.
- Lựa chọn là 5: kết thúc chương trình.



Bài tập



- Bài tập 6.6 (*):

  Viết các lệnh biên dịch cho những dự án có tổ chức như sau:

Dự án đơn nhiều thư mục con	**Dự án đơn dùng thư viện ngoài**	**Dự án phức dự án 1 dùng dự án 2**
project/ bin/ src/ subfolder/	project/ bin/ lib/ libA/ libB/ src/ subfolder/	project/ lib/ libA/ libB/ subproject1/ bin/ src/ subfolder1/ subproject2/ bin/ src/ subfolder2/