KỸ THUẬT LẬP TRÌNH

BÀI 5

KỸ THUẬT XỬ LÝ

CHUỖI KÝ TỰ - CON TRỎ - TỆP TIN

BÀI 5. KỸ THUẬT XỬ LÝ CHUỖI KÝ TỰ - CON TRỎ - TỆP TIN

- PHẦN 1: CHUỖI KÝ TỰ
- PHẦN 2: CON TRỎ
- PHẦN 3: TỆP TIN

- **Các thao tác cơ bản**

[1]. Khai báo biến kiểu chuỗi ký tự

Cách 1: Mảng các ký tự

char S[200];

- Ta thu được biến S có thể chứa tối đa 200 ký tự
- Ký tự kết thúc chuỗi '\n' luôn được thêm vào cuối S.

Cách 2: Kiểu string

string S;

- Ta thu được biến S có thể chứa các chuỗi ký tự có độ dài rất lớn
- Biến S kiểu string được thao tác theo kiểu **hướng đối tượng**.

- **Các thao tác cơ bản**

[2]. Nhập chuỗi ký tự

Cách 1: Sử dụng gets

```
fflush(stdin); gets(S);
```

Cách 2: Sử dụng cin

```
cin.getline(S, 30);
```

- cin.getline(S, P); được sử dụng để nhập xâu ký tự kiểu mảng char

- **Các thao tác cơ bản**

[3]. Duyệt chuỗi ký tự

- Lấy độ dài của chuỗi ký tự S:

```
strlen(S)
```

- Duyệt chuỗi:

```
for(int i = 0; i < strlen(S); i++)
//Thăm S[i]
```

- **Các thao tác đặc thù**

[1]. Phép gán chuỗi

- Phép gán thông thường '=' là không hợp lệ: ~~**S = "HA NOI";**~~
- Hàm **strcpy()** copy chuỗi: Lệnh sau gán chuỗi ký tự P sang biến S

strcpy(S, P);

S là một biến chuỗi ký tự kiểu mảng char (ví dụ char S[200];).
P là một biến chuỗi ký tự kiểu mảng char hoặc một hằng chuỗi ký tự.

strcpy(S, "HA NOI");

- **Các thao tác đặc thù**

[2]. Phép so sánh hai chuỗi

- Phép so sánh thông thường '==' là không hợp lệ:

~~**if (S == "HA NOI")**~~

- Hàm **strcmp()** so sánh hai chuỗi: Hàm trả về 0 nếu hai chuỗi bằng nhau, ngược lại, hàm trả về giá trị khác 0:

if(strcmp(S, P) == 0)

S, P là các biến chuỗi ký tự kiểu mảng char hoặc một hằng chuỗi ký tự. Ví dụ: **Nếu S bằng "HA NOI"**:

if(strcmp(S, "HA NOI") == 0)

- **Các thao tác đặc thù**

[3]. Thao tác với mã ASCII:

- Mỗi ký tự tương đương với một số nguyên, là mã ASCII của nó

  Ví dụ: Ký tự 'A' có mã 65. Ký tự 'a' có mã 97.

- Khi so sánh các ký tự, thực chất là so sánh các mã ASCII của nó

  Ví dụ: 'A' < 'B' vì mã của 'B' là 66.

- Lấy mã ASCII của một ký tự hoặc ngược lại: Ép kiểu

  Ví dụ: **cout<<(int) 'A';** sẽ in ra màn hình mã của ký tự 'A'

  cout<<(char) 65; sẽ in ra màn hình ký tự 'A'

- **Các bài toán cơ bản trên chuỗi**

[1]. Các bài toán sắp xếp tìm kiếm

[2]. Các bài toán thống kê trên xâu: Thống kê số ký tự, số từ, số câu…

[3]. Bài toán chuẩn hóa xâu

[4]. Bài toán tách, chèn, xóa

🕮 BÀI TẬP 5.1

- Nhập 1 chuỗi ký tự, cho biết chuỗi có bao nhiêu từ

(một từ là đoạn ký tự liên tiếp, dài nhất không có dấu cách)

Ví dụ: Chuỗi : **"HA NOI NGAY THANG NAM"** có 5 từ

Chuỗi : **"_ _HA_ _NOI_NGAY_ _ _THANG_NAM_ _"** cũng có 5 từ

☞ **Phương pháp: Đếm số chỗ bắt đầu của một từ**

Bắt đầu của một từ là khi: Có một ký tự trống theo sau là một ký tự khác trống

"_ _HA_ _NOI_NGAY_ _ _THANG_NAM_ _"

Chú ý: Nếu ký tự đầu tiên không phải là ký tự trống, ta đã đếm thiếu từ đầu tiên

🕮 BÀI TẬP 5.1

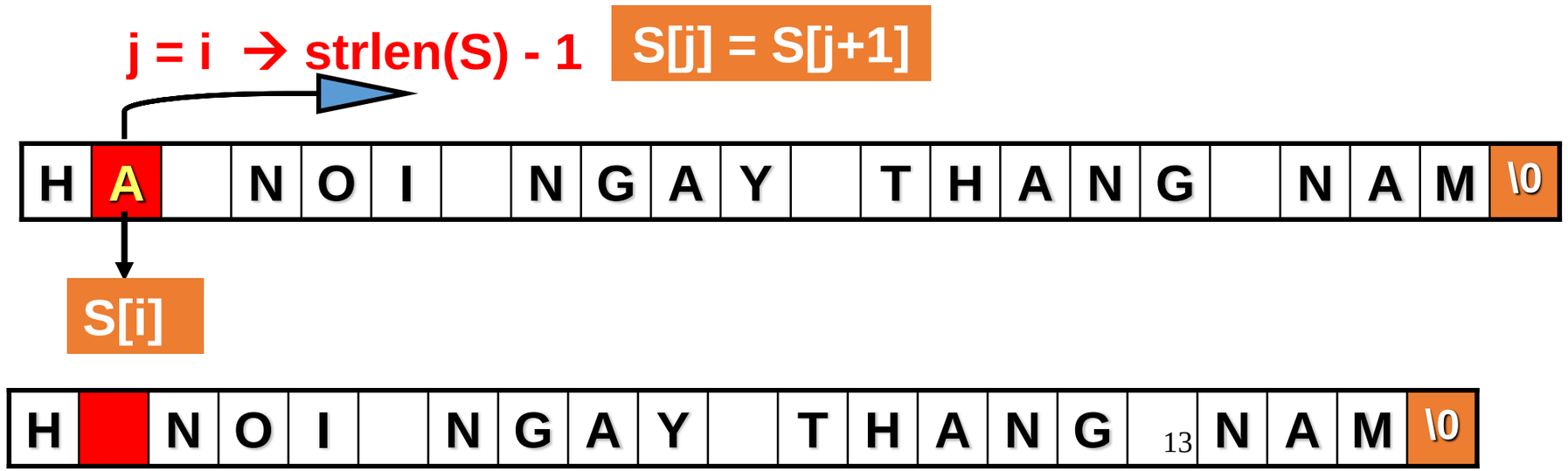
```
int countword(char S[])
{
      int d=0;
      for(int i=0; i<strlen(S)-1; i++)
            if(S[i] == ' ' && S[i+1] != ' ')
                  d++;
      if(S[0] != ' ')
            d++;
      return d;
}
```

🕮 BÀI TẬP 5.2

- Nhập 1 chuỗi ký tự S và một ký tự C. Hãy xóa mọi ký tự C trong S

Ví dụ: S: **"HA NOI NGAY THANG NAM"** và C = 'A'. Kết quả sau khi xóa:

S: **"H NOI NGY THNG NM"**



🕮 BÀI TẬP 5.2

- Nhập 1 chuỗi ký tự S và một ký tự C. Hãy xóa mọi ký tự C trong S

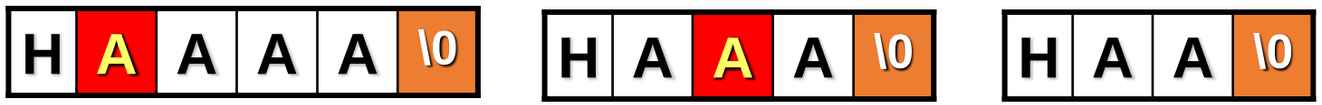
Phương pháp:

- Duyệt xâu S: **for(int i = 0; i < strlen(S); i++)**
- Nếu gặp ký tự C thì: **if(S[i] == C)**
- Xóa S[i]:
  - ➢ Đẩy toàn bộ các ký tự sau S[i] lên trước một vị trí (lưu ý đẩy cả ký tự kết thúc chuỗi).

🕮 BÀI TẬP 5.2

```
void deleteall(char S[], char C)
{
      for(int i=0; i<strlen(S); i++)
      if(S[i] == C)
            for(int j=i; j<strlen(S); j++)
                  S[j] = S[j+1];
}
```

- Trường hợp ký tự C xuất hiện liên tiếp: có thể bỏ sót



🕮 BÀI TẬP 5.2

```
void deleteall(char S[], char C)
{
      for(int i=0; i<strlen(S); i++)
      while(S[i] == C)
            for(int j=i; j<strlen(S); j++)
                  S[j] = S[j+1];
}
```

🕮 BÀI TẬP 5.3

- Cho một biểu thức toán học với các dấu mở/đóng ngặc '(' và ')' dưới dạng một chuỗi ký tự. Các dấu mở/đóng ngoặc được gọi là hợp lệ nếu nó được đặt đúng chỗ trong biểu thức toán học đó.
- Ví dụ biểu thức: (a+b) * (c+d) có các dấu mở/đóng ngoặc hợp lệ, nhưng biểu thức: (a+b) * (c+d)) hoặc (a+b)) * ((c+d) lại không hợp lệ.
- Hãy cho biết một biểu thức có các dấu mở/đóng ngoặc hợp lệ hay không.

🕮 BÀI TẬP 5.3

- Sử dụng hai biến đếm d1 và d2 để đếm số dấu mở và đóng ngoặc.
- Điều kiện hợp lệ:

  Mở trước đóng: nếu đếm từ trái qua phải, d1 luôn $\geq$ d2

  Mở bằng đóng: kết thúc quá trình đếm, d1 = d2

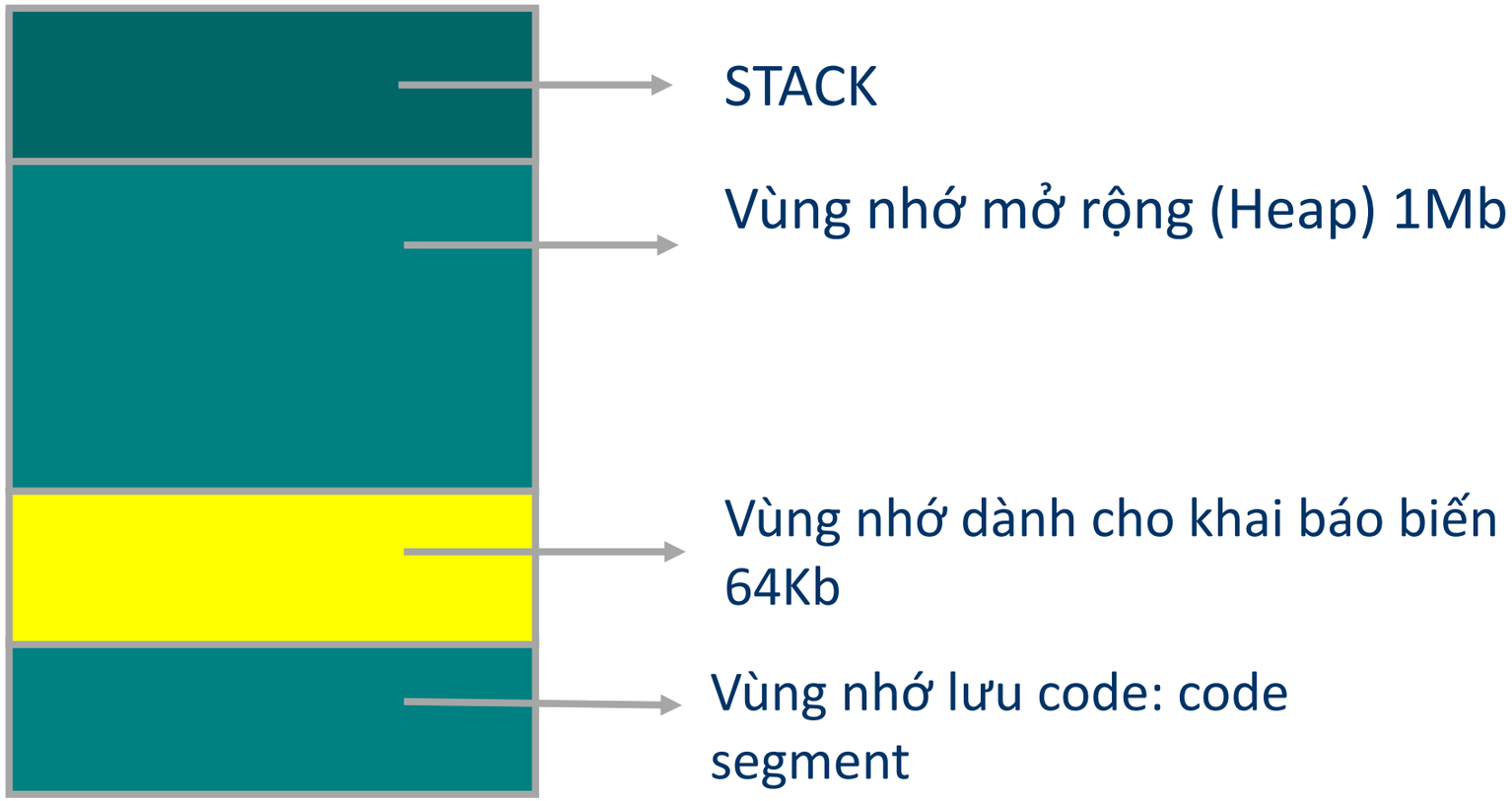
🕮 BÀI TẬP 5.3

```
bool isInvalid(char S[])
{
    int d1=0, d2=0;
    for(int i=0; i<strlen(S); i++)
    {
        if( S[i] == '(' ) d1++;
        if( S[i] == ')' ) d2++;
        if(d1<d2) return false;
    }
    if(d1 != d2) return false;
    return true;
}
```

- **Kỹ thuật lập trình với con trỏ**
  - Khái niệm
  - Các thao tác cơ bản trên con trỏ
  - Cấp phát – thu hồi bộ nhớ
  - Con trỏ và hàm
  - Con trỏ và mảng

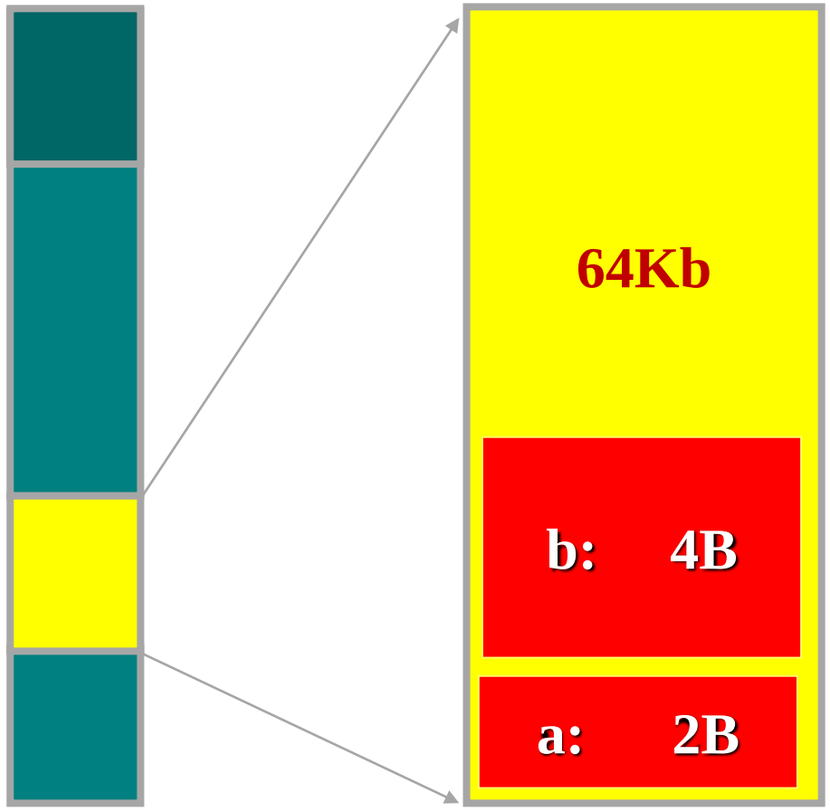
Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Khái niệm con trỏ**
  - **Cấu trúc bộ nhớ**



Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Khái niệm con trỏ**
  - **Địa chỉ của biến**



- Mỗi Byte trong bộ nhớ đều được đánh địa chỉ, là một con số hệ 16
- Địa chỉ của biến là địa chỉ của Byte đầu tiên trong ô nhớ dành cho biến

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Khái niệm con trỏ**
  - **Phép lấy địa chỉ của biến**

☞ **Ba yếu tố của biến**

[1]. Tên biến: **a**

[2]. Giá trị của biến: **a, 5**

[3]. Địa chỉ của biến: **&a, FFF1**

int a = 5;

☞ **Phép lấy địa chỉ của biến:**

& ⟨Tên biến⟩

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Khái niệm con trỏ**
  - **Lưu trữ địa chỉ của biến**

☞ **Biến thông thường không thể lưu trữ địa chỉ của biến khác**

```
int a = 5;
```

☒ `int b = &a;`

☞ **Lưu trữ địa chỉ của biến: Sử dụng biến đặc biệt**

```
int a = 5;
```

☑ `int * c = &a;`

☞ **Biến c được gọi là biến con trỏ hay con trỏ !**

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Khái niệm con trỏ**

Con trỏ: là một biến đặc biệt,
dùng để chứa ĐỊA CHỈ của biến khác, cùng kiểu

☞ **Hãy so sánh** một biến nguyên thông thường và một biến con trỏ nguyên

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Các thao tác cơ bản trên con trỏ**

1. Khai báo biến con trỏ

⟨Kiểu⟩ * ⟨Tên⟩;

Ví dụ: int * p;

2. Trỏ tới một biến cùng kiểu

⟨Con trỏ⟩ = & ⟨Biến⟩:

Ví dụ:

```
int a = 5;
int * b = &a;  //Con trỏ b trỏ tới biến a
```

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Các thao tác cơ bản trên con trỏ**

3. Truy xuất tới biến

int a = 5; int *P = &a;

☞ Khi một con trỏ đang trỏ tới một biến, ta có thể sử dụng con trỏ để truy xuất tới biến đó.

☞ ***(P)** chính là tên gọi khác của biến mà P đang trỏ tới (biến **a**)

***P = 10;**	↔	**a = 10;**

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Các thao tác cơ bản trên con trỏ**

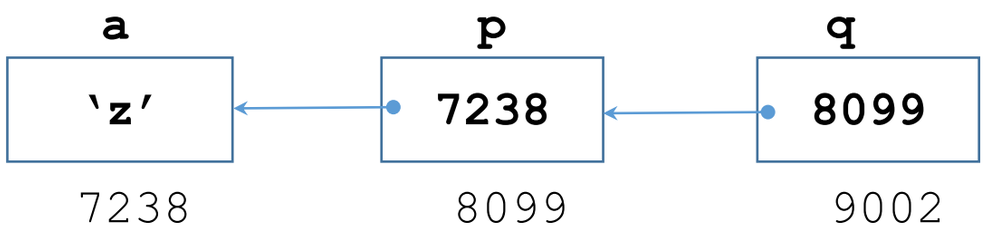
4. Trỏ tới con trỏ

- Con trỏ trỏ tới các con trỏ khác cùng kiểu

```
char a;    char * p;     char ** q;
a = 'z';   p = &a;       q = &p;
```

q là con trỏ trỏ tới con trỏ.

- Để sử dụng **q** truy xuất tới dữ liệu trong **a**, ta viết: ****q**



Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Cấp phát/thu hồi bộ nhớ cho con trỏ**

1. Cấp phát bộ nhớ

- Con trỏ có thể quản lý một dãy liên tiếp các ô nhớ khi ta cấp phát cho nó

⟨Con_trỏ⟩ = new ⟨Kiểu⟩ [⟨n⟩];

- Ví dụ:

```
int * p = new int;
```


```
int * p = new int[3];
```

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Cấp phát/thu hồi bộ nhớ cho con trỏ**

2. Thu hồi bộ nhớ

delete[] ⟨Con_trỏ⟩;

Ví dụ: **delete[] p;**

🕗 Hàm **delete[]** thu hồi bộ nhớ đã cấp cho con trỏ. Nó không làm thay đổi giá trị của con trỏ (là một địa chỉ). Nó chỉ đơn giản cho hệ thống quản lý bộ nhớ biết rằng bit bộ nhớ tại địa chỉ đó, địa chỉ đã được dự trữ trước đó, không còn được dự trữ nữa và có thể sử dụng lại.

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Cấp phát/thu hồi bộ nhớ cho con trỏ**

3. malloc/ calloc/ realloc

⟨Con_trỏ⟩ = (⟨Kiểu*⟩) malloc(⟨Size⟩);

⟨Con_trỏ⟩ = (⟨Kiểu*⟩) calloc(⟨n⟩, sizeof(Kiểu);

⟨Con_trỏ⟩ = (⟨Kiểu*⟩) realloc(⟨Con_trỏ⟩, ⟨New_size⟩;

- Thu hồi bộ nhớ: **free(⟨Con_trỏ⟩);**

🕗 **realloc()** được dùng để cấp phát lại bộ nhớ mà trước đó ta đã cấp phát bằng malloc/ calloc. Trong trường hợp bộ nhớ trước đó được cấp phát bằng new, ta không sử dụng realloc.

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Cấp phát/thu hồi bộ nhớ cho con trỏ**

4. Cấp phát bộ nhớ động (Dynamic memory allocation)

- **Cấp phát bộ nhớ động:** Là việc thực hiện cấp phát bộ nhớ linh hoạt theo cách thủ công bởi lập trình viên.
- Bộ nhớ được cấp phát động được cấp phát trên Heap hoặc Stack.

Ví dụ: **int a[100];**

```
int *a = new int[100];
....
delete[] a;
```

🕐 Đối với các biến bình thường như “int a[100]”, v.v., bộ nhớ được cấp phát và phân bổ tự động.

🕐 Đối với bộ nhớ được cấp phát động như “int * a = new int [100]”, lập trình viên có trách nhiệm thu hồi bộ nhớ khi không còn cần thiết nữa. Nếu lập trình viên không thu hồi bộ nhớ, nó sẽ gây ra rò rỉ bộ nhớ (memory leak)

Khái niệm
Thao tác cơ bản
Con trỏ và hàm
Cấp phát bộ nhớ
Con trỏ và mảng
Bài tập

- **Con trỏ và hàm**
  - **Con trỏ trỏ tới hàm**

- Xét hai hàm đơn giản thực hiện phép cộng và phép trừ:

```
int cong(int a, int b)
{
      return a+b;
}

int tru(int a, int b)
{
      return a-b;
}
```

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Con trỏ và hàm**
  - **Con trỏ trỏ tới hàm**
    - Hàm **tinhtoan** sau có thể thực hiện phép cộng hay phép trừ tùy ý. Hàm sử dụng con trỏ **p** để trỏ tới hàm **cong, tru**:

```
int tinhtoan(int x, int y, int (*p)(int,int))
{
    return (*p)(x,y);
}
int main()
{
    int m = tinhtoan(7, 5, cong);
    int n = tinhtoan(2, 9, tru);
    cout <<n<<", "<<m;
}
```

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Con trỏ và mảng**
  - **Tên mảng chính là con trỏ**
    - Khi khai báo một mảng, tên mảng chính là con trỏ, trỏ tới phần tử đầu tiên của mảng

int a[5];

- Cộng trừ địa chỉ:

*(a) *(a+1) *(a+2) *(a+3) *(a+4)

a[i] ↔ *(a+i)

Khái niệm
Thao tác cơ bản
Con trỏ và hàm
Cấp phát bộ nhớ
Con trỏ và mảng
Bài tập

- **Con trỏ và mảng**
  - **Con trỏ + Cấp phát bộ nhớ = Mảng**
    - Khi khai báo một con trỏ, ta cần cấp phát bộ nhớ cho nó để biến nó thành mảng

int *a;

a = new int[5];

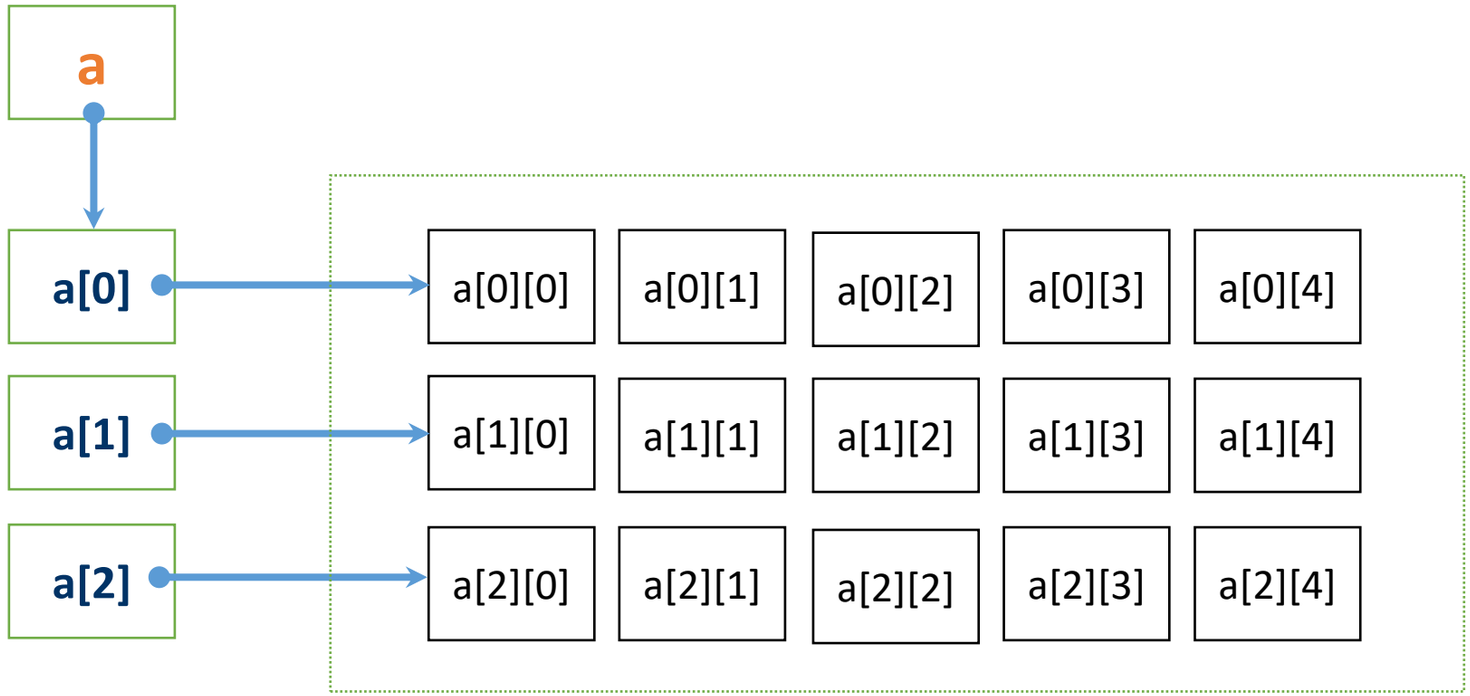
***(a)** ***(a+1)** ***(a+2)** ***(a+3)** ***(a+4)**

☞ Hãy so sánh:

int a[5];	**int *a = new int[5];**

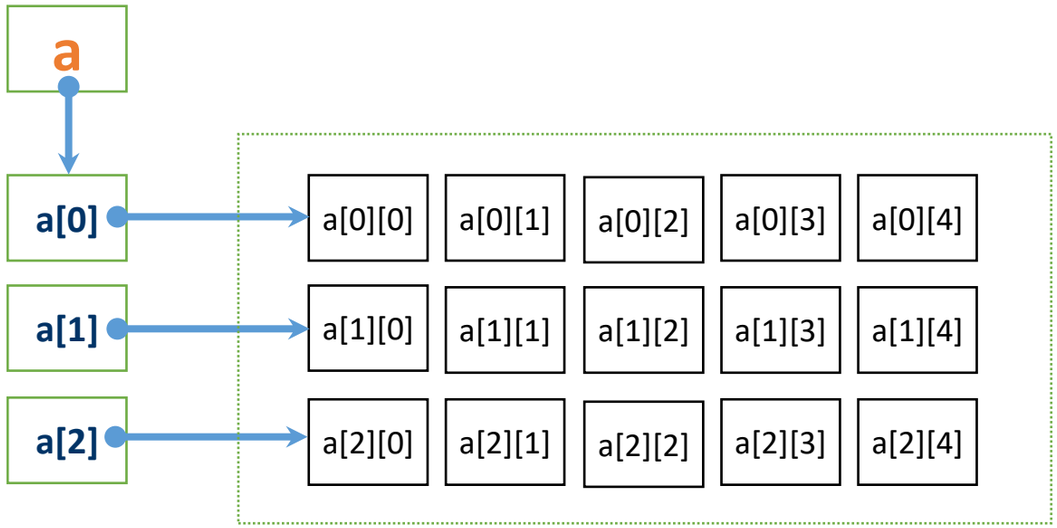
Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Con trỏ và mảng**
  - **Con trỏ và mảng hai chiều**
    - **Mảng hai chiều:** Mảng của các mảng một chiều.



Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Con trỏ và mảng**
  - **Con trỏ và mảng hai chiều**



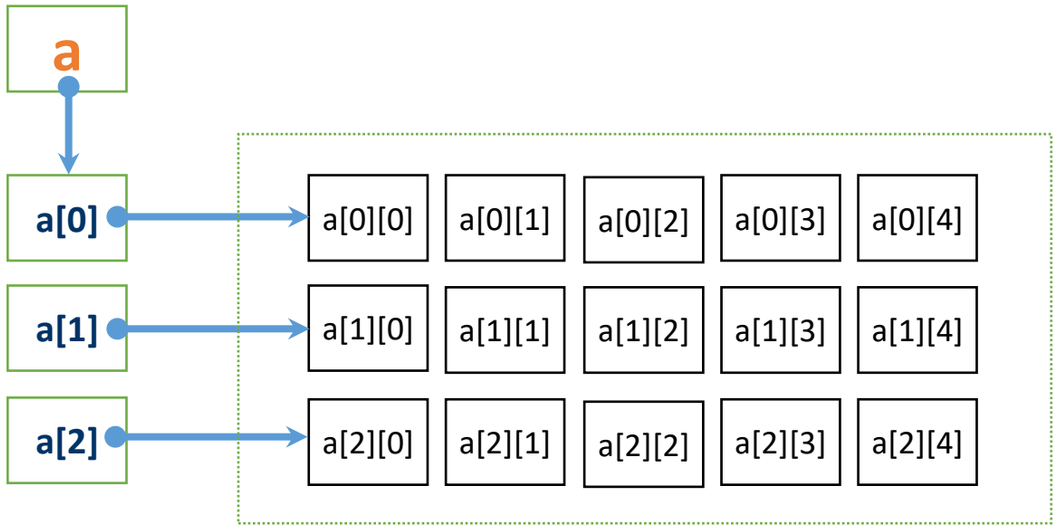
- **a[0], a[1], a[2]:** Là ba con trỏ quản lý ba mảng một chiều
- **a:** Là con trỏ, trỏ tới con trỏ (a[0]). Do vậy, ta khai báo:

⟨Kiểu⟩ ** ⟨Tên⟩;

Ví dụ: int **a; float ** b;

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

- **Con trỏ và mảng**
  - **Con trỏ và mảng hai chiều**
    - **Cấp phát bộ nhớ cho mảng hai chiều:**



Đoạn chương trình sau cấp phát bộ nhớ cho một mảng hai chiều **a(n × m)** phần tử **thực**

```
a = new float*[n];
for(int i=0; i<n; i++)
        a[i] = new float[m];
```

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

🕮 BÀI TẬP 5.4

- Dùng con trỏ cấp phát bộ nhớ động để: Nhập vào một mảng a gồm n phần tử nguyên. Sao chép các phần tử chẵn của mảng đặt vào cuối mảng. In kết quả ra màn hình.

Khái niệm
Thao tác cơ bản
Cấp phát bộ nhớ
Con trỏ và hàm
Con trỏ và mảng
Bài tập

🕮 BÀI TẬP 5.5

- Dùng con trỏ cấp phát bộ nhớ động để: Nhập một mảng a gồm n phần tử thực. Tách các phần tử âm sang mảng b và các phần tử dương sang mảng c. In ba mảng a, b, c ra màn hình.

🕮 BÀI TẬP 5.6

- Dùng con trỏ cấp phát bộ nhớ động để: Nhập vào một mảng a(n×m) phần tử thực.
- In mảng vừa nhập và các giá trị lớn nhất trên từng dòng của mảng ra màn hình.

- **Kỹ thuật xử lý tệp văn bản**
  - **Mở/đóng tệp**
  - **Ghi dữ liệu vào tệp**
  - **Đọc dữ liệu từ tệp**

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Mở/ đóng tệp**
  - **#include fstream**
  - Mở tệp để ghi dữ liệu:

    Ghi đè: **ofstream f(⟨Tên_tệp⟩, ios::out);**

    Ghi bổ sung: **ofstream f(⟨Tên_tệp⟩, ios::app);**
  - Mở tệp để đọc dữ liệu: **ifstream f(⟨Tên_tệp⟩, ios::in);**
  - Đóng tệp: **f.close();**

☞ f được gọi là dòng nhập/ xuất tệp hay đơn giản là biến tệp. Bạn có thể đặt tên tùy ý, không nhất thiết là f.

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Ghi dữ liệu vào tệp**

f<<⟨Dữ_liệu⟩;

Trong đó ⟨Dữ_liệu⟩ có thể là:

- **Hằng, biến, hàm, biểu thức**

```
f<<“HA NOI NGAY ”<<20<<“ THANG”<<7;
f<<(a+b+sqrt(X))/2;
```

- **Cờ định dạng, xuống dòng**

```
f<<setw(3)<<HOTEN<<endl;
```

☞ Lệnh xuất dữ liệu vào tệp tương tự lệnh xuất dữ liệu ra màn hình. Nhưng thay vì dung cout (dòng xuất màn hình) ta sử dụng f (dòng xuất tệp !

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

BÀI TẬP 5.7

- Nhập một mảng a gồm n phần tử nguyên từ bàn phím. Ghi dữ liệu của a vào tệp.
- Nhập một ma trận b(nxm) gồm các phần tử thực từ bàn phím. Ghi dữ liệu của b vào tệp theo quy định: dòng đầu tiên ghi **n m**; các dòng tiếp theo ghi các dòng của ma trận.

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Đọc dữ liệu vào tệp**
  - **Đọc từng dòng của tệp**

f.getline(⟨Biến⟩, ⟨Số_ký_tự_tối_đa⟩);

- Ví dụ:

```
char S[200];
f.getline(S, 200);
```

- Đọc toàn bộ tệp với getline(): Đoạn chương trình sau đây đọc toàn bộ tệp dữ liệu và in ra màn hình.

```
char S[200];
while(!f.eof())
{
        f.getline(S, 200);    cout<<S<<endl;
}
```

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Đọc dữ liệu vào tệp**
  - **Đọc từng dòng của tệp**

f.getline(⟨Biến⟩, ⟨Số_ký_tự_tối_đa⟩);

✍ Lệnh đọc **f.getline()** tương tự như lệnh nhập **cin.getline().** Điểm khác biệt là ở chỗ: **cin.getline()** đọc dữ liệu từ bàn phím.

✍ Đọc từng dòng của tệp thích hợp khi nộp dung của tệp là văn bản và ta chỉ coi dữ liệu đọc được là các dòng văn bản.

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Đọc dữ liệu vào tệp**
  - **Đọc từng cụm trong tệp**

✍ Một cụm dữ liệu trong tệp được định nghĩa là một dãy liên tiếp các ký tự không chứa dấu cách

✍ Đọc từng cụm thích hợp với tệp dữ liệu kiểu số, khi mà mỗi cụm là một con số và ta dễ dàng chuyển đổi kiểu dữ liệu cũng như lưu trữ nó.

f>>⟨Biến⟩;

✍ Lệnh đọc cụm **f>>⟨Biến⟩;** tương tự lệnh **cin>>⟨Biến⟩;**. Điểm khác biệt là ở chỗ lệnh cin đọc dữ liệu từ bàn phím.

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Đọc dữ liệu vào tệp**
  - **Đọc từng cụm trong tệp**

- Giả sử ta có tệp dữ liệu số như hình sau (tệp VIDU1.txt)

5	3	1	2	4	3	2

- Đoạn chương trình đọc toàn bộ tệp lên một biến mảng một chiều a kích thước n và in mảng ra màn hình:

```
ifstream f("D:/VIDU1.txt", ios::in);
int n=0;  int a[100];
while(!f.eof())
{
        f>>a[n];        n++;
}
f.close();
```

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Đọc dữ liệu vào tệp**
  - **Đọc từng cụm trong tệp**

✍ Việc không biết trước cấu trúc của tệp dữ liệu sẽ gây khó khăn cho lập trình viên khi đọc:

- **Ví dụ:** Đọc toàn bộ tệp dữ liệu VIDU2.txt đang chứa một mảng hai chiều **n × m** phần tử lên các biến **a, n, m**

5	3	1	2
4	3	2	3
4	3	3	4

Do không biết trước kích thước của mảng đang chứa trong tệp, ta sẽ gặp khó khăn khi cấp phát bộ nhớ cho a.

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Đọc dữ liệu vào tệp**
  - **Đọc từng cụm trong tệp**
    - Để đơn giản, ta thêm hai giá trị n, m vào dòng đầu tiên của tệp

3	**4**		
5	3	1	2
4	3	2	3
4	3	3	4

- Việc đọc tệp bây giờ trở nên dễ dàng !

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

- **Đọc dữ liệu vào tệp**
  - **Đọc từng cụm trong tệp**

```
ifstream f("D:/VIDU.txt", ios::in);
int n, m;  int **a;
f>>n>>m;
a = new int*[n];
for(int i=0; i<n; i++)
    a[i] = new int[m];
```

```
for(int i=0; i<n; i++)
for(int j=0; j<m; j++)
{
    f>>a[i][j];
}
f.close();
```

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

🕮 BÀI TẬP 5.8

- Tương tự như bài tập 5.7: Nhập một ma trận b(nxm) gồm các phần tử thực từ bàn phím. Ghi dữ liệu của b vào tệp theo quy định: dòng đầu tiên ghi **n m**; các dòng tiếp theo ghi các dòng của ma trận.
- Mở tệp vừa tạo, đọc toàn bộ nội dung của tệp và in ra màn hình theo đúng định dạng (không quan tâm tới giá trị đọc được).
- Mở tệp vừa tạo, đọc toàn bộ nội dung của tệp ra các biên p, q, a(p×q). In ma trận a ra màn hình sau khi đọc được.

Mở/ đóng tệp
Ghi tệp
Đọc tệp
Bài tập

🕮 BÀI TẬP 5.9

- Cho bộ dữ liệu theo đường link sau (sinh viên tải bộ dữ liệu về máy tính).
- Đọc dữ liệu từ bộ dữ liệu lưu vào các biến n, m, a(n×m) tương ứng
- Tính trung bình cộng từng cột trong ma trận a vừa đọc được.

KẾT THÚC