BAN HỌC TẬP

UIT
Trường Đại học
Công nghệ Thông tin

Khoa Khoa học
và Kỹ thuật Thông tin

Nhập môn lập trình

Trainer: - Phạm Quốc Cường (CNCL2020)
- Nguyễn Thiện Thuật (KHDL2020)
- Đặng Thị Thúy Hồng (KHDL 2020)

Mảng

Mảng

Khái niệm

Là một dãy các phần tử có cùng kiểu dữ liệu, mỗi phần tử trong mảng biểu diễn một giá trị.

Kính thước được xác định ngay khi khai báo và không đổi.

Phân loại:

- Mảng 1 chiều
- Mảng 2 chiều

BAN HỌC TẬP

Mảng 1 chiều

Khai báo

<tên kiểu> <tên mảng>[số phần tử];

- **Ví dụ:** int a[1000];

<tên kiểu> <tên mảng>[số phần tử]={dãy giá trị};

- **Ví dụ:** int x[2]={1, 2, 3};

<tên kiểu> <tên mảng>[]={dãy giá trị};

- **Ví dụ:** char s[]={a, b, c};

Mảng 1 chiều

Chỉ số và cách truy cập

Chỉ số mảng là một số nguyên không âm

Chỉ số thuộc đoạn [0;n-1] (với n là số phần tử)

Truy cập các phần tử mảng thông qua chỉ số:

<Tên mảng>[<chỉ số mảng>]

- **Ví dụ:** int a[3]={30, 3, 2021};
  Đúng: a[0], a[1],a[2]
  Sai: a[-1], a[3], a[4]
  giá trị các phần tử trong mảng: a[0]=30; a[1]=3; a[2]=2021;

Các thuật toán mảng 1 chiều

Nhập mảng

```
void Nhapmang(int a[], int n)
{
   for (int i=0; i < n; i++)
          cin >> a[i];
}
```

Xuất mảng

```
void Xuatmang(int a[], int n)
{
   for (int i=0; i < n; i++)
          cout << a[i] << " ";
}
```


BAN HỌC TẬP

Các thuật toán mảng 1 chiều

Sắp xếp

```
void swap(int &a, int &b)
{
   int temp = a;
   b = a;
   a = temp;
}
```

```
void BubbleSort(int a[], int n)
{
   for (int i=0; i < n; i++)
         for (int j=i+1; j < n; j++)
               if (a[i] > a [j])
               swap(a[i], a[j]);
}
// Sap xep tang
```

Các thuật toán mảng 1 chiều

Đếm số nguyên tố

```
#include <math.h>
bool checkNguyenTo(int x)
{
   if (x < 2) return false;
   for (int i=2; i <= sqrt(x); i++)
        if(x % i == 0) return false;
   return true;
}
```

```
int demNguyenTo(int a[], int n)
{
   int dem = 0;
   for (int i =0; i < n; i++)
          if (checkNguyenTo(a[i]))
                 dem++;
   return dem;
}
```


BAN HỌC TẬP

Mảng 2 chiều

Khai báo

<tên kiểu> <tên mảng> [<số hàng>][<số cột>];

- Ví dụ: int a[50][100]; foat[10][10];

Chỉ số và truyền mảng

	0	1	2
0	3	4	2
1	8	9	7
2	1	5	4

Các thao tác truyền mảng 2 chiều cơ bản giống với với mảng 1 chiều, nhưng thay vì dùng một vòng lặp thì chúng ta sẽ dùng hai vòng lặp

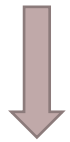
Các thuật toán mảng 2 chiều

Xóa hàng của ma trận

```
void Tong(int a[][Max], int x, int n)
{
for(int i=x; i < n; i++)
      for(int j=0; j < n; j++)
            a[i][j] = a[i+1][j];
}
```

x = 2
n = 3

7	7	4	9
2	0	0	2
1	5	0	5
9	9	9	9



7	7	4	9
2	0	0	2
9	9	9	9

BAN HỌC TẬP

Kiểu cấu trúc

Kiểu cấu trúc

Khái niệm

Kiểu cấu trúc (struct) là một tập hợp của những kiểu dữ liệu khác nhau được gộp lại tạo thành một kiểu dữ liệu mới.

Cú Pháp

struct <tên kiểu cấu trúc> {

<kiểu dữ liệu> <tên biến 1>;

...

<kiểu dữ liệu> <tên biến n>;

};

struct <tên kiểu cấu trúc > <tên biến>; //C++ có thể bỏ qua

```
struct PhanSo
{
Int TuSo;
Int MauSo;
};
struct PhanSo PSa, PSb;
```


BAN HỌC TẬP

Kiểu cấu trúc

Cú Pháp sử dụng typedef

typedef struct

{

<kiểu dữ liệu> <tên biến 1>;

...

<kiểu dữ liệu> <tên biến n>;

} <tên kiểu cấu trúc > ;

<tên kiểu cấu trúc > <tên biến>;

.

```
typedef struct
{
Int TuSo;
Int MauSo;
} PhanSo;
PhanSo PSa, PSb;
```

Khi dùng typedef **không** có từ khóa struct lúc khai báo biến.

Kiểu cấu trúc

Khởi tạo cho biến cấu trúc

struct <tên kiểu cấu trúc>

{

<kiểu dữ liệu> <tên biến 1>;

…

<kiểu dữ liệu> <tên biến n>;

} <tên biến> = {<giá trị 1>,… ,};

Ví dụ:

```
Struct PhanSo
{
int TuSo;
int MauSo;
};
PhanSo PSa = {3,4};
```


BAN HỌC TẬP

Kiểu cấu trúc

Truy xuất dữ liệu kiểu cấu trúc

Đặc điểm

- Không thể truy xuất trực tiếp
- Thông qua toán tử thành phần cấu trúc hay còn gọi là toán tử chấm

Cú Pháp: <tên biến cấu trúc>.<tên thành phần>

Ví dụ:

```
struct PhanSo {
        int TuSo, MauSo;
};
PhanSo PSa = {3,4};
cout << PSa.TuSo << "/" << PSa.MauSo ;
```

Output: 3/4

BAN HỌC TẬP

Kiểu cấu trúc

Gán dữ liệu kiểu cấu trúc

Có 2 cách:

Gán trực tiếp	<biến cấu trúc đích> = <biến cấu trúc nguồn>;
	PhanSo PSa = {3,4}; PhanSo PSa = PSb;
Gán từng thành phần	<biến cấu trúc đích>.<tên thành phần> = <biến cấu trúc nguồn>;
	PhanSo PSa = {3,4}; PhanSo PSb; PSa.TuSo = PSb.TuSo; PSa.MauSo = PSb.MauSo * 2;

Cấu trúc phức tạp

Thành phần của cấu trúc là cấu trúc khác nhau

Ví Dụ:

```
struct DIEM
{
        int x;
        int y;
};
```

```
struct HINHCHUNHAT
{
        struct DIEM traitren;
        struct DIEM phaiduoi;
} hcn1;
…
hcn1.traitren.x = 2912;
hcn1.traitren.y = 1706;
```

Mảng cấu trúc

Tương tự như mảng với kiểu dữ liệu cơ sở (char, int, float, ...)

Ví Dụ:

```
struct DIEM
{
        int x;
        int y;
};

DIEM mang1[20];
DIEM mang2[10] = {{3, 2}, {4, 4}, {2, 7}};
```

Truyền cấu trúc cho hàm

Giống như truyền kiểu dữ liệu cơ sở

- Tham trị (không thay đổi sau khi kết thúc hàm)
- Tham chiếu
- Con trỏ

```
void Nhap(PhanSo &PS)
{
cout << “Nhap Tu So: “;
cin >> PS.TuSo;
cout << “Nhap Mau So: “;
Cin >> PS.MauSo;
}
```

```
double GiaTriPS(PhanSo PS)
{
double x;
x = (double)PS.TuSo / PS.MauSo;
return x;
}
```

Con trỏ

Con trỏ ?? – Nỗi sợ ??

1. Con trỏ là gì mà khiến nhiều người sợ hãi khi nhắc tới nó vậy kìa =(
2. Con trỏ có ăn được không ??

BAN HỌC TẬP

Con trỏ - Pointer

1. Con trỏ là gì?

– **Con trỏ** là một biến bình thường nhưng có thể trỏ đi lung tung trong bộ nhớ. **Và giá trị của con trỏ là địa chỉ**

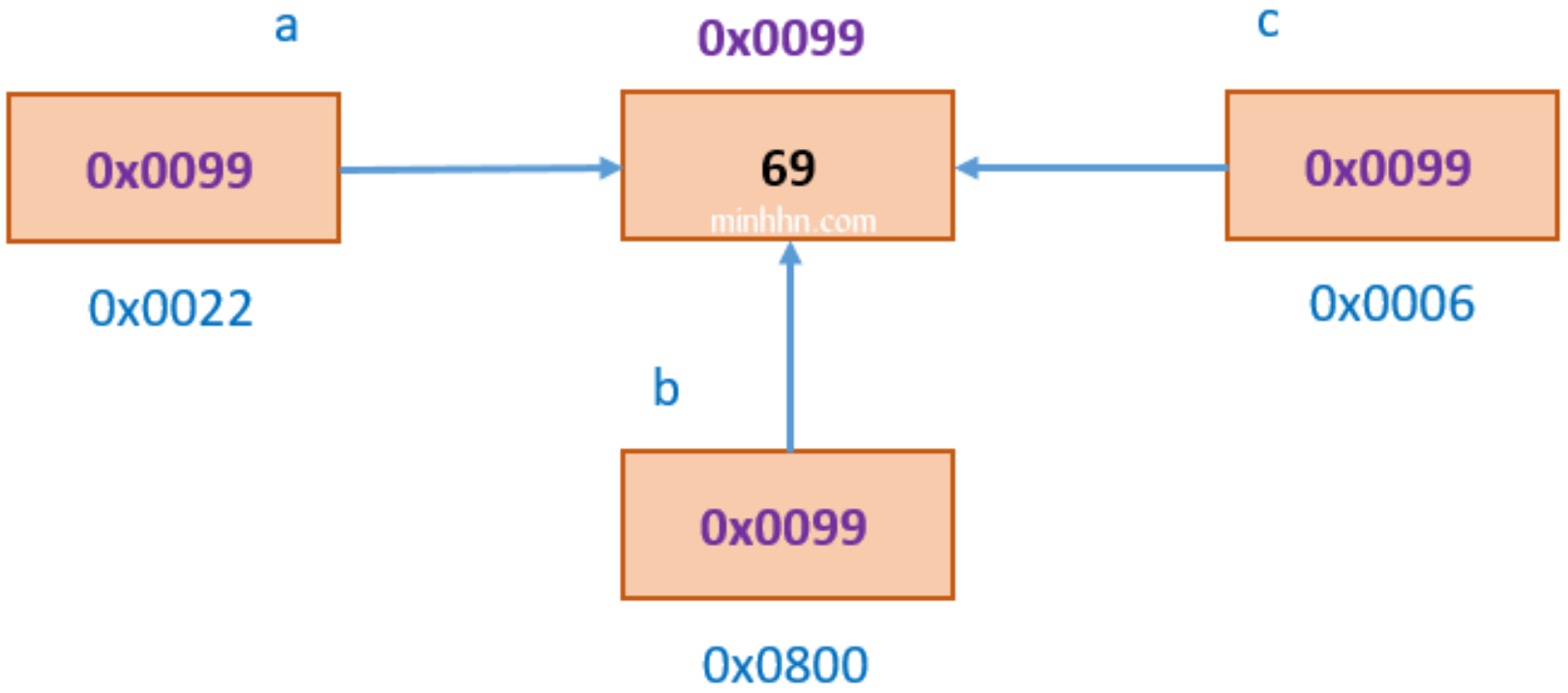
– Về **bản chất con trỏ cũng như một biến bình thường**: nó cũng có tên biến, giá trị của biến, địa chỉ của biến. Nhưng có điểm khác là:

+ Những biến bình thường thì nó chỉ nằm cố định trong 1 ô nhớ, còn biến con trỏ thì nó còn có thể trỏ đến các ô nhớ khác nhau.

+ <Kiểu dữ liệu> khi khai báo cho con trỏ không phải là kiểu dữ liệu của nó, mà là kiểu dữ liệu của vùng nhớ mà nó đang trỏ đến.

Con trỏ - Pointer

Ta có 3 biến con trỏ a, b, c có địa chỉ lần lượt là 0x0022, 0x0800, 0x0006 trỏ đến vùng nhớ có địa chỉ 0x0099 lưu giá trị 69



BAN HỌC TẬP

Con trỏ - Pointer

2. Tại sao lại có con trỏ? Sử dụng để làm gì?

– Vì biến con trỏ có thể trỏ đi lung tung trong bộ nhớ nên việc sử dụng bộ nhớ sẽ linh hoạt hơn.

– Áp dụng cho mảng động. Có nghĩa là khi chúng ta sử dụng mảng tĩnh với số lượng phần tử của mảng là cố định, chẳng hạn như mảng có 100 phần tử, thì dù chúng ta chỉ sử dụng 5 – 10 phần tử để thao tác tính toán thôi, thì bộ nhớ cũng sẽ cấp phát 100 ô nhớ, do đó mà sẽ gây ra lãng phí bộ nhớ không đáng có. Còn khi chúng ta sử dụng mảng động dùng con trỏ thì chúng ta sử dụng 5 thì bộ nhớ cấp phát 5 ô nhớ, sử dụng 10 thì bộ nhớ cấp phát 10 ô nhớ.

Con trỏ - Pointer

3. Cách khai báo con trỏ

– Cú pháp:

<Kiểu dữ liệu> *<Tên của con trỏ>;

– Trong đó:

<Kiểu dữ liệu> : Bao gồm các kiểu dữ liệu có sẵn (int, float, double, char, void) và kiểu dữ liệu do người dùng tự định nghĩa (Book, HocSinh, PhanSo,…)

Dấu * : biểu thị đây là biến con trỏ.

<Tên của con trỏ> : Tuân theo quy tắc đặt tên biến trong lập trình.

Con trỏ - Pointer

- Các con trỏ cùng kiểu có thể gán cho nhau.
- Dùng **toán tử *** để **lấy giá trị tại địa chỉ mà con trỏ trỏ đến** (và toán tử * cũng thể hiện cho một biến là con trỏ, nên các bạn khi sử dụng cần phân biệt rõ ràng là khi nào *p là con trỏ p, khi nào *p là đang muốn lấy giá trị của con trỏ p).
- Dùng **toán tử &** để lấy địa chỉ của một biến.
- Con trỏ khai báo kiểu int thì chỉ trỏ đến ô nhớ kiểu int, con trỏ khai báo kiểu float thì chỉ trỏ được đến ô nhớ có kiểu là float,… Nếu con trỏ khai báo kiểu int mà chúng ta cho trỏ đến ô nhớ kiểu float hoặc ngược lại thì sẽ lỗi.

Con trỏ - Pointer

Ví dụ: Cho các khai báo

Int x, y, *px, *c;

Khai báo 2 biến x, y có kiểu int và 2 con trỏ px, c có kiểu int

Float *t, *d;

Khai báo 2 con trỏ t, d có kiểu float.

c = &y;

Gán giá trị địa chỉ biến y cho con trỏ c

px = &x;

Gán giá trị địa chỉ biến x cho con trỏ px

Tương tự, ta hiểu câu lệnh **t = &y** là gì?

!!!!!! Đây là câu lệnh sai vì t là con trỏ kiểu float, nó chỉ có thể chứa địa chỉ của biến float.

Qui tắc khi dùng con trỏ

Cách 1: Dùng tên con trỏ

- Lưu ý: Giá trị của biến con trỏ là địa chỉ! Vì vậy, khi muốn gán giá trị cho con trỏ phải dung toán tử & để lấy địa chỉ!
- Ví dụ:

```
float a, *p, *q;
p = &a;
q = p;
```

- Sau khi thực hiện xong, biến q sẽ chứa giá trị là địa chỉ của biến a.
- Cũng giống các biến khác, giá trị con trỏ cũng có thể thay đổi. Chẳng hạn: nếu con trỏ p chứa địa chỉ của mảng a[i] (p=a) thì khi thực hiện p=p+1 nó sẽ chứa địa chỉ của phần tử a[i+1]

Qui tắc khi dùng con trỏ

Cách 2: Dùng dạng khai báo của con trỏ

- Lưu ý: Giá trị của biến con trỏ là địa chỉ! Khi muốn lấy giá trị của biến con trỏ được trỏ đến, ta dùng toán tử *;
- Ví dụ:

```
float a = 5, *p;
p = &a;
Cout << *p;
```

- Sau khi thực hiện xong, chương trình in ra giá trị 5.
- Điều này được hiểu là *p và a là tương đương nhau trong mọi ngữ cảnh

Hàm có đối số là con trỏ

```
void hoanvi (int *x, int *y) {
        float z = *x;
        *x = y;
        *y = z;
}
int main() {
        int x = 5, y = 7;
        hoanvi(&x, &y);
        cout << x << "\t" << y;
}
```

Kết quả sau khi thực hiện???

- Chương trình báo lỗi biên dịch.
- Lý do:

  Giá trị của *x có kiểu là int

  Giá trị của y là địa chỉ

⇒ Không thể thực hiện phép gán *x = y;

⇒ Sửa thành: *x = *y;

BAN HỌC TẬP

Hàm có đối số là con trỏ

```
void hoanvi (int *x, int *y) {
        float z = *x;
        *x = *y;
        *y = z;
}
int main() {
        int x = 5, y = 7;
        hoanvi(&x, &y);
        cout << x << "\t" << y;
}
```

```
void hoanvi (int &x, int &y) {
        float z = x;
        x = y;
        y = z;
}
int main() {
        int x = 5, y = 7;
        hoanvi(x, y);
        cout << x << "\t" << y;
}
```

Các phép toán số học con trỏ

Phép toán tính khoảng cách giữa 2 con trỏ

<kiểu dữ liệu> *p1, *p2;

p1 – p2 cho ta khoảng cách (theo số phần tử) giữa hai con trỏ (cùng kiểu)

Các phép toán so sánh

Phép so sánh: So sánh địa chỉ giữa hai con trỏ (thứ tự ô nhớ)

==, !=, >, >=, <, <=

Không thể thực hiện các phép toán: * / %

BAN HỌC TẬP

Con trỏ & mảng 1 chiều

- Trong C/C++ có mối **quan hệ chặt chẽ** giữa **con trỏ & mảng**, các **phần tử của mảng có thể xác định nhờ chỉ số (*a[i]*) hoặc con trỏ *(*(p+i))***
- Giả sử, ta có khai báo: **Double a[20]**
- Khi đó, **phép toán &a[i] (hoặc a+i) với i trong đoạn [0,19] cho giá trị là địa chỉ của phần tử a[i]**
- Các lệnh có ý nghĩa tương đương:

  a tương đương với &a[0]

  a+i tương đương với &a[i]

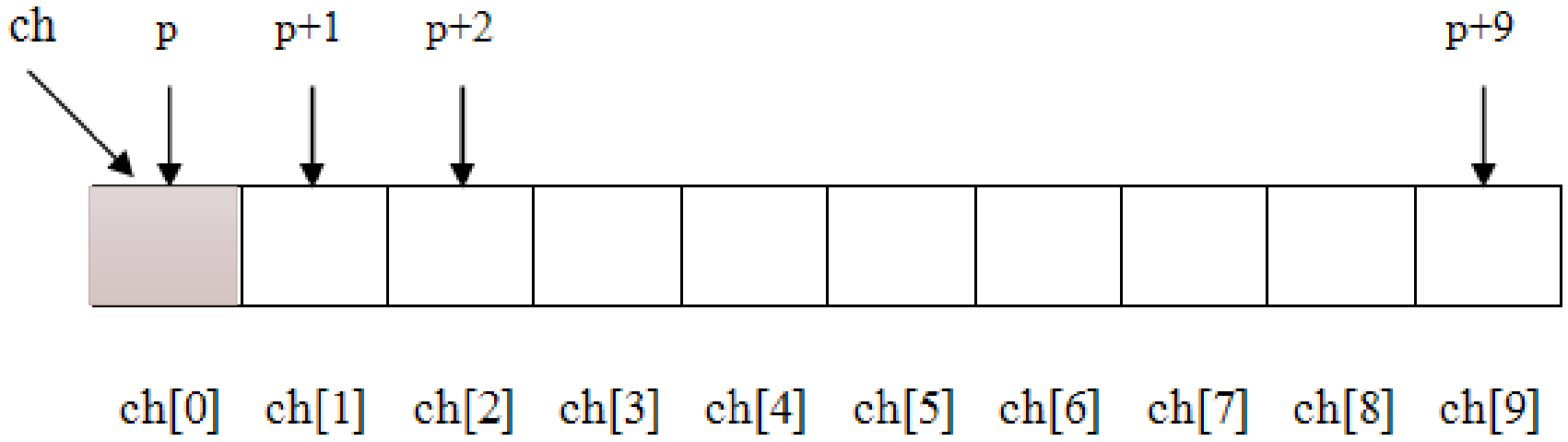
  ***(a+i) tương đương với a[i]**

BAN HỌC TẬP

Con trỏ & mảng 1 chiều

Ví dụ:

```
char ch[10], *p;
p = ch;
```



BAN HỌC TẬP

Cấp phát động

- Cấp phát bộ nhớ tĩnh (static memory allocation)

Khai báo biến, cấu trúc, mảng, …

Bắt buộc phải biết trước cần bao nhiều bộ nhớ lưu trữ tốn bộ nhớ, không thay đổi được kích thước, …

- Cấp phát động (dynamic memory allocation)

Cần bao nhiêu cấp phát bấy nhiêu.

Có thể giải phóng nếu không cần sử dụng.

Cấp phát động

- Có thể cấp phát động cho biến con trỏ bằng **toán tử new.**
- Toán tử new sẽ **tạo ra biến "không tên" cho con trỏ trỏ tới**.
- Cú pháp: <type> *<pointerName> = new <type>

Ví dụ: int *ptr = new int;

Tạo ra một biến "không tên" và gán ptr trỏ tới nó

Có thể làm việc với biến "không tên" thông qua *ptr

Cấp phát động

- Toán tử delete dùng để giải phóng vùng nhớ do con trỏ trỏ tới (con trỏ được cấp pháp bằng toán tử new).
- Cú pháp: delete <pointerName>;

!!!!! Ghi chú: Sau khi gọi toán tử delete thì con trỏ vẫn trỏ tới vùng nhớ trước khi gọi hàm delete. Ta gọi là "con trỏ lạc". Ta vẫn có thể gọi tham chiếu trên con trỏ, tuy nhiên:

- Kết quả không lường trước được
- Thường là nguy hiểm
- Hãy tránh con trỏ lạc bằng cách gán con trỏ bằng NULL sau khi delete.

Ví dụ:

- delete pointer;
- pointer = NULL;

Cấp phát động

```
int main() {
   int arr[10];
   return 0;
}
```

```
int main() {
   int *arr;
   arr = new int[10];
   return 0;
}
```


BAN HỌC TẬP

Mảng, con trỏ & xâu kí tự

```
char *st;
gets(st);
cout << st;
```

Input: “Ban hoc tap”

Kết quả sau khi thực hiện???

- Chương trình báo **lỗi biên dịch.**
- Lý do:

Câu lệnh gets(st) là đúng nhưng không thể thực hiện do vùng nhớ mà con trỏ st trỏ tới chưa được xác định.

⇒ Chẳng hạn, ta tạo 1 con trỏ vô danh bằng toán tử new

⇒ **st = new char();**

BAN HỌC TẬP

Mảng, con trỏ & xâu kí tự

```
char *st, t[15];
t = "Ban hoc tap";
cout << t;
```

Kết quả sau khi thực hiện???

Cách chữa lại:

```
char *st, t[15];
st = t;
strcpy(st, “Ban hoc tap”);
cout << t;
```

Tuy nhiên, đoạn chương trình dưới đây lại hợp lệ

```
char *st, t[15];
st = "Ban hoc tap";
cout << st;
```


BAN HỌC TẬP

Mảng, con trỏ & xâu kí tự

```
char *st;
st = "Ban hoc tap";
cout << st;
cout << st+1;
cout << st[1];
cout << *(st+1);
```

Kết quả sau khi thực hiện???

St ~ st+0;

St[1] ~ *(st+1);

St+1 sẽ bỏ qua phần tử thứ 0

BAN HỌC TẬP

Con trỏ

Xem đoạn chương trình dưới đây và cho biết kết quả

```
void hamf(int* a) {
  a = new int[5];
  for (int i = 0; i < 5; i++)
        a[i] = i + 1;
}
void main() {
  int n = 5;
  int* a = &n;
  cout << "Giá trị * a = "<<*a;
  hamf(a);
  cout << "Giá trị * a = "<<*a;
}
```

```
void hamf(int*& a) {
  a = new int[5];
  for (int i = 0; i < 5; i++)
        a[i] = i + 1;
}
void main() {
  int n = 5;
  int* a = &n;
  cout << "Giá trị * a = "<<*a;
  hamf(a);
  cout << "Giá trị * a = "<<*a;
}
```


BAN HỌC TẬP

Xâu kí tự

Xâu kí tự

1. Cách khai báo xâu kí tự

– Cú pháp:

char*<Tên của xâu>;

char <Tên của xâu> [số_phần tử];

– Trong đó:

Dấu * : biểu thị đây là biến con trỏ.

<Tên của xâu> : Tuân theo quy tắc đặt tên biến trong lập trình.

Xâu kí tự

Sau 1 thời gian soạn slide, mình không biết cách nào là khai báo đúng cho xâu kí tự, các bạn giúp mình với =(

```
char* str = ‘BHT’;
char str[3] = “BHT”;
char str = “B”;
char str[1] = “”;
char s[3] = { 'B','H','T' };
```

Có bao nhiêu cách khai báo hợp lệ taaa?

BAN HỌC TẬP

Xâu kí tự

Chuỗi là một mảng ký tự được kết thúc bằng ký tự null (‘\0’).

Ký tự null (‘\0’) là ký tự dùng để kết thúc Chuỗi

Hằng Chuỗi là Chuỗi được bao quanh bởi cặp dấu nháy đôi.

Ví dụ: “Hello”

Ví dụ: để khai báo một mảng str chứa chuỗi có độ dài 20 ký tự, ta khai báo:

```
char str[21];
```

Xâu kí tự

Ví dụ xâu kí tự: “**Ban hoc tap KH&KTTT**”

Ta nhận thấy có khoảng cách giữa các từ, mà thông thường chúng ta sử dụng “cin” thì chương trình sẽ hiểu kí tự kết thúc tại dấu “ “... Vậy làm sao để nhập cả xâu trên bây giờ taaa?

Ta sử dụng hàm:

gets(xâu_kí_tự);

gets_s(xâu_kí_tự, số_phần_tử);

getline(cin, xâu_kí_tự);

BAN HỌC TẬP

Xâu kí tự

Các hàm cần biết

Một số hàm thuộc thư viện <string.h>

- strlen: hàm tính độ dài chuỗi ký tự
- strcpy: hàm sao chép chuỗi ký tự
- strlwr/strupr: hàm chuyển chuỗi thành chuỗi viết thường / hoa
- strcmp : hàm so sánh 2 chuỗi có phân biệt hoa thường
- stricmp : hàm so sánh 2 chuỗi không phân biệt hoa thường
- strcat : hàm nối 2 chuỗi
- strstr : hàm tìm chuỗi trong chuỗi