

# CON TRỎ

KỸ THUẬT LẬP TRÌNH

GVHD: Trương Toàn Thịnh

# NỘI DUNG

- Dữ liệu có cấu trúc
- Cấu trúc hóa vùng nhớ thô
- Toán tử chỉ số biến cấu trúc
- Mảng cấu trúc
- Các kỹ thuật nâng cao

# DỮ LIỆU CÓ CẤU TRÚC

- Trong thực tế dữ liệu thường theo cấu trúc
- Dùng từ khóa `struct` kết hợp `typedef` để định nghĩa các kiểu dữ liệu mới
- Dùng toán tử ‘.’ hoặc ‘->’ để truy xuất tới các thành phần bên trong
- Dùng toán tử `sizeof` kết hợp `pragma pack(1)` để xác định chính xác kích thước kiểu dữ liệu mới
- Dùng từ khóa `union` nếu muốn các thành phần dùng chung vùng nhớ

# DỮ LIỆU CÓ CẤU TRÚC (KÍCH THƯỚC)

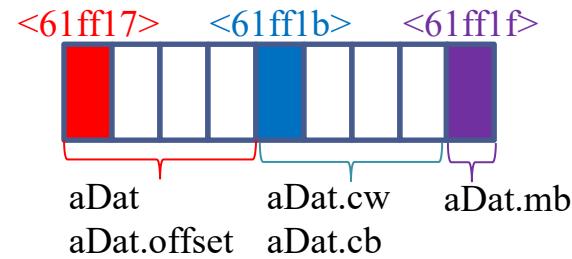
- Ví dụ (cần dùng **pragma pack(1)**) sizeof = 9
  - **typedef struct {**
    - long offset; 4
    - union{ unsigned long cw; unsigned char cb[4];}; 4
    - char mb; 1
  - **} HeadStruct;**
  - **typedef struct { long x, y; } Point;**
  - **typedef struct {**
    - short nVers;
    - Point Vers[1];
  - **} Polygon;**
  - **typedef struct {**
    - short nVers;
    - Point\* Vers;
  - **} PolygonP;**

```
void main(){  
    HeadStruct aDat;  
    HeadStruct* pDat = &aDat;  
}  
  
sizeof(aDat): 9  
sizeof(pDat): 4  
sizeof(HeadStruct): 9  
sizeof(Point): 8  
sizeof(Polygon): 10  
sizeof(PolygonP): 6
```

# DỮ LIỆU CÓ CẤU TRÚC (ĐỊA CHỈ CÁC THÀNH PHẦN)

- Ví dụ (cần dùng **pragma pack(1)**)
  - **typedef struct {**
    - long offset;
    - union{ unsigned long cw; unsigned char cb[4];};
    - char mb;
  - **} HeadStruct;**
  - **typedef struct { long x, y; } Point;**
  - **typedef struct {**
    - short nVers;
    - Point Vers[1];
  - **} Polygon;**
  - **typedef struct {**
    - short nVers;
    - Point\* Vers;
  - **} PolygonP;**

```
void main(){  
    HeadStruct aDat;  
    HeadStruct* pDat = &aDat;  
}  
  
&aDat: 61ff17  
&aDat.offset: 61ff17  
&aDat.cw: 61ff1b  
&aDat.cb: 61ff1b  
&aDat.mb: 61ff1f
```

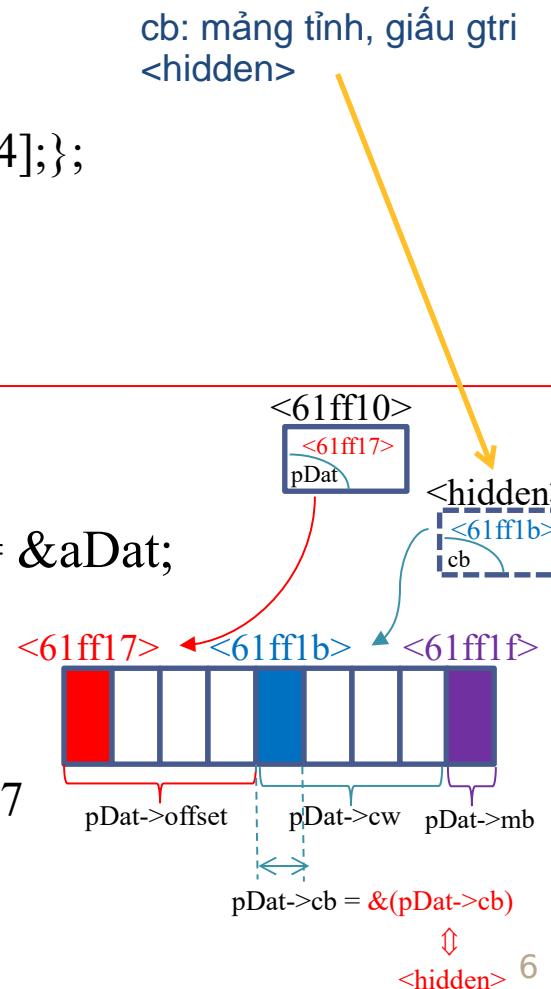


# DỮ LIỆU CÓ CẤU TRÚC (ĐỊA CHỈ CÁC THÀNH PHẦN)

- Ví dụ (cần dùng pragma pack(1))

```
◦ typedef struct {  
    • long offset;  
    • union{ unsigned long cw; unsigned char cb[4];};  
    • char mb;  
◦ } HeadStruct;  
◦ typedef struct { long x, y;} Point;  
◦ typedef struct{  
    • short nVers;  
    • Point Vers[1];  
◦ } Polygon;  
◦ typedef struct{  
    • short nVers;  
    • Point* Vers;  
◦ } PolygonP;
```

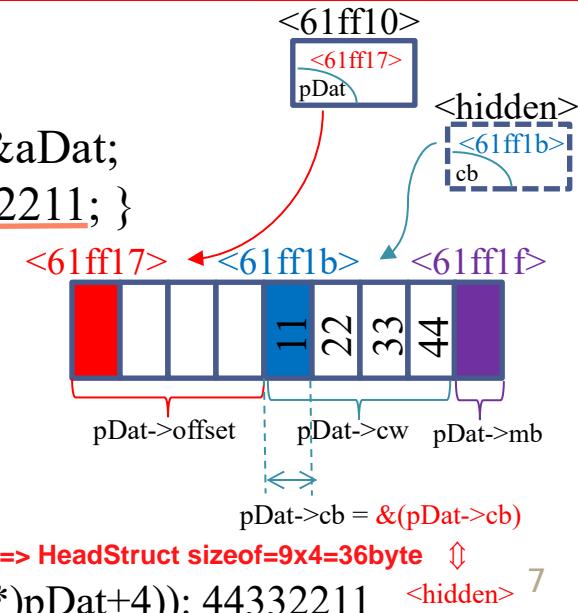
```
void main(){  
    HeadStruct aDat;  
    HeadStruct* pDat = &aDat;  
}  
&pDat: 61ff10  
pDat: 61ff17  
&(pDat->offset): 61ff17  
pDat->cb: 61ff1b  
&(pDat->cw): 61ff1b  
&(pDat->mb): 61ff1f
```



# DỮ LIỆU CÓ CẤU TRÚC (GIÁ TRỊ VÙNG UNION)

- Ví dụ (cần dùng pragma pack(1))
  - `typedef struct {`
    - `long offset;`
    - `union{ unsigned long cw; unsigned char cb[4];};`
    - `char mb;`
  - `} HeadStruct;`
  - `typedef struct { long x, y;} Point;`
  - `typedef struct{`
    - `short nVers;`
    - `Point Vers[1];`
  - `} Polygon;`
  - `typedef struct{`
    - `short nVers;`
    - `Point* Vers;`
  - `} PolygonP;`

```
void main(){  
    HeadStruct aDat;  
    HeadStruct* pDat = &aDat;  
    (*pDat).cw = 0x44332211; }  
  
*(pDat->cb+0): 11  
pDat->cb[1]: 22  
*(pDat->cb+2): 33  
pDat->cb[3]: 44  
pDat->cw: 44332211  
*(pDat+4): 44332211 Sais=> HeadStruct sizeof=9x4=36byte  
*((unsigned long*)((char*)pDat+4)): 44332211 <hidden>
```



# DỮ LIỆU CÓ CẤU TRÚC (KỸ THUẬT KHỎI GÁN)

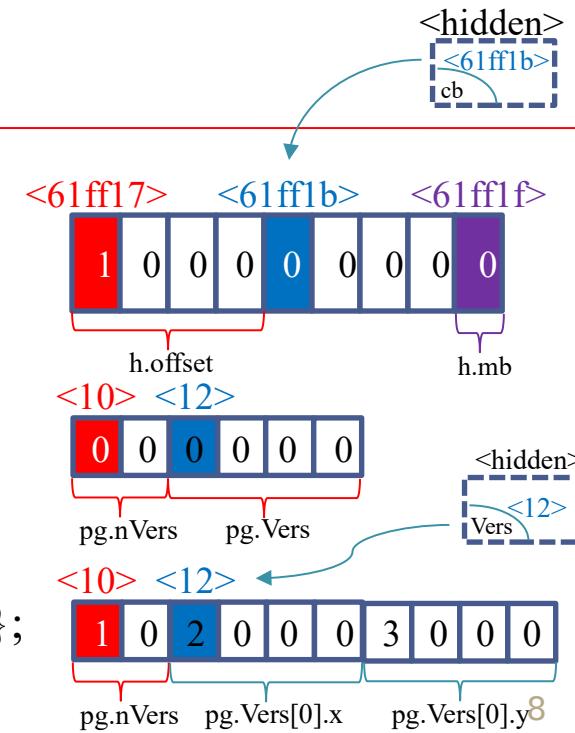
- Ví dụ (cần dùng **pragma pack(1)**)

```
◦ typedef struct {  
    • long offset;  
    • union{ unsigned long cw; unsigned char cb[4];};  
    • char mb;  
◦ } HeadStruct;  
◦ typedef struct { long x, y;} Point;  
◦ typedef struct{  
    • short nVers;  
    • Point Vers[1];  
◦ } Polygon;  
◦ typedef struct{  
    • short nVers;  
    • Point* Vers;  
◦ } PolygonP;
```

HeadStruct h = {1};

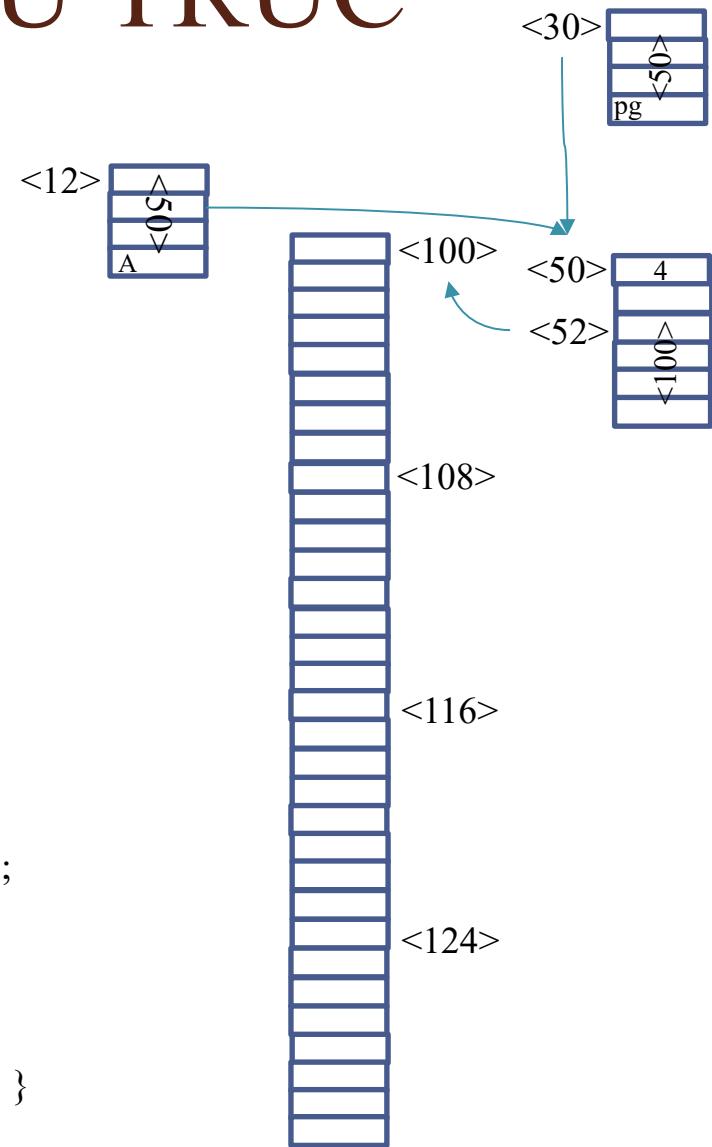
PolygonP pg = {0};

Polygon pg = {1, {2, 3}};



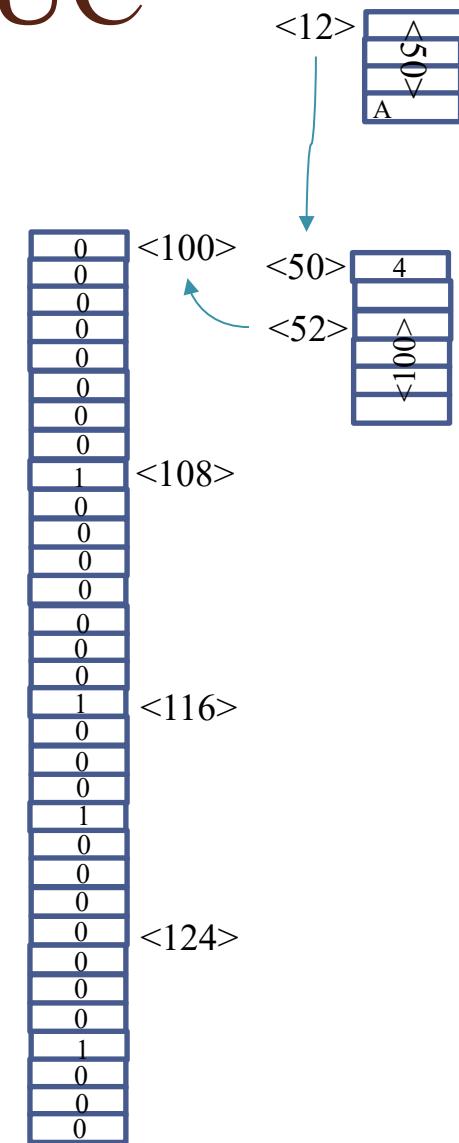
# DỮ LIỆU CÓ CẤU TRÚC

- Ví dụ với PolygonP (cách 1)
  - `typedef struct { long x, y; } Point;`
  - `typedef struct{`
    - `short nVers; Point* Vers;`
  - `} PolygonP;`
- `void main(){`
  - `Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};`
  - `PolygonP* A = PgAllocA(4);`
- `}`
- `PolygonP* PgAllocA(int n){`
  - `if(n < 0) return NULL;`
  - `int szHead = sizeof(short) + sizeof(Point*);  
2+4 = 6 byte`
  - `PolygonP* pg = (PolygonP*)calloc(szHead, 1);`
  - `if(pg == NULL) return NULL;`
  - `pg->nVers = n;`
  - `pg->Vers = (Point*)calloc(n, sizeof(Point));`
  - `if(pg->Vers == NULL){ free(pg); pg = NULL; }`
  - `return pg;`
- `}`



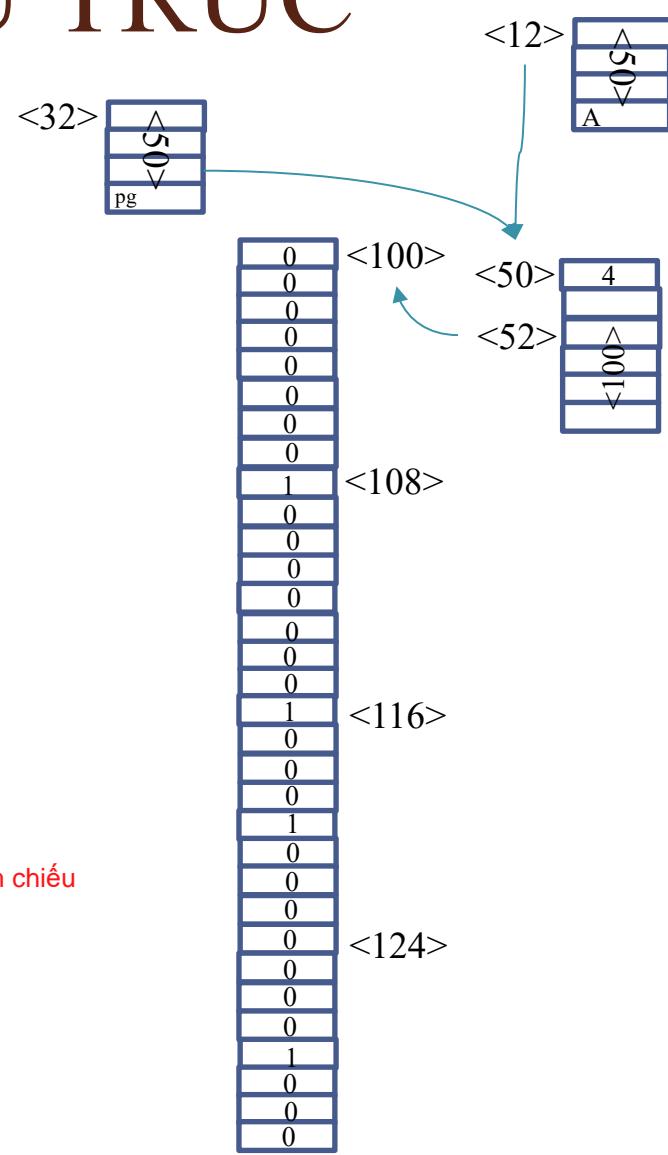
# DỮ LIỆU CÓ CẤU TRÚC

- Ví dụ với PolygonP (cách 1)
  - **typedef struct {**
    - long x, y;
  - } Point;
  - **typedef struct {**
    - short nVers; Point\* Vers;
  - } PolygonP;
  - **void main(){**
    - Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};
    - PolygonP\* A = PgAllocA(4);
    - **if(A != NULL){**
      - **for(int i = 0; i < 4; i++){**
        - A->Vers[i] = P[i];
      - }
    - }
  - }



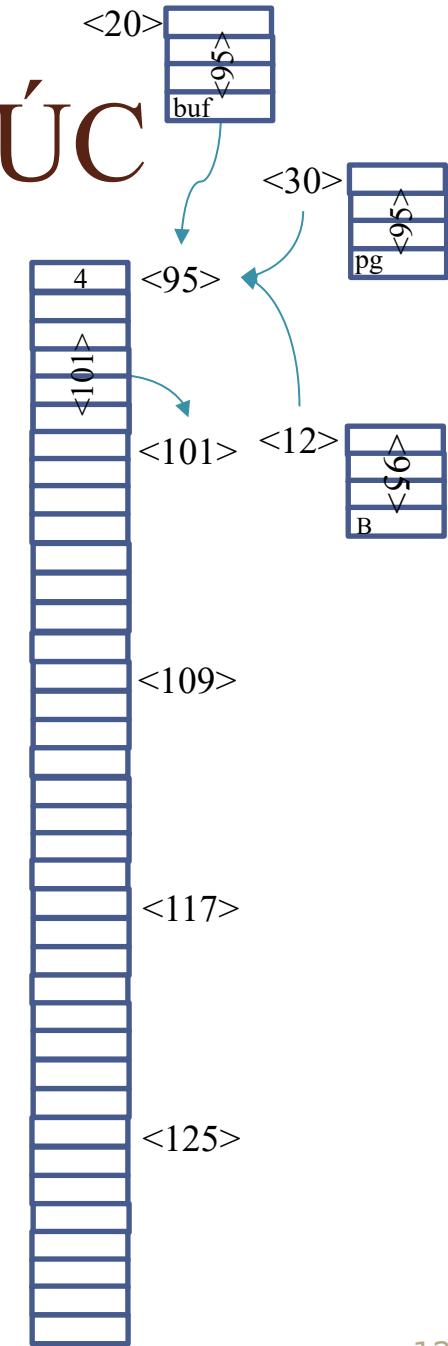
# DỮ LIỆU CÓ CẤU TRÚC

- Ví dụ với PolygonP (cách 1)
  - `typedef struct { long x, y; } Point;`
  - `typedef struct{`
    - `short nVers; Point* Vers;`
  - `} PolygonP;`
  - `void main(){`
    - `Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};`
    - `PolygonP* A = PgAllocA(4);`
    - `if(A != NULL){`
      - `for(int i = 0; i < 4; i++) A->Vers[i] = P[i];`
      - `pgFreeA(A);`
    - `}`
  - `}` trỏ cấp 1 / \*& trỏ cấp 2 dạng tham chiếu
  - `void pgFreeA(PolygonP* pg){`
    - `if(pg != NULL){`
      - `if(pg->Vers != NULL) free(pg->Vers);`
      - `free(pg);`
    - `}`
  - `}`



# DỮ LIỆU CÓ CẤU TRÚC

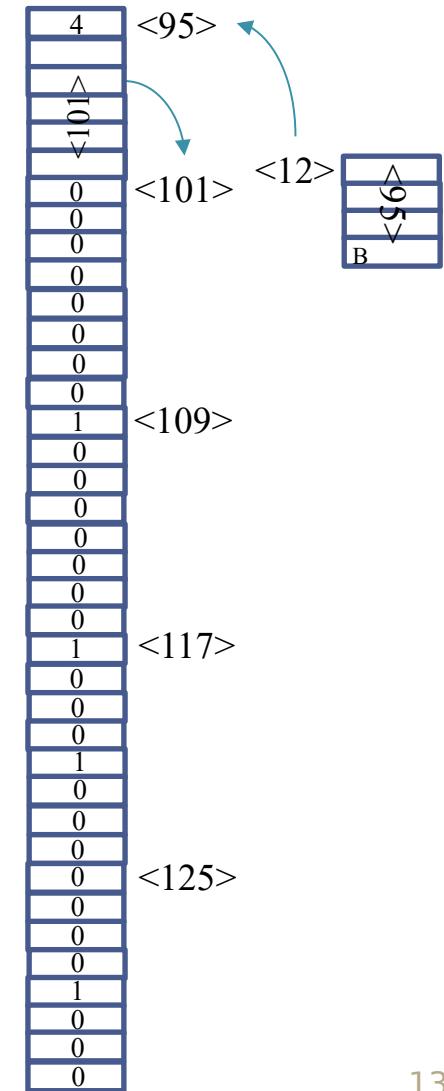
- Ví dụ với PolygonP (cách 2)
  - `typedef struct { long x, y; } Point;`
  - `typedef struct{`
    - `short nVers; Point* Vers;`
  - `} PolygonP;`
  - `void main(){`
    - `Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};`
    - `PolygonP* B = PgAllocB(4);`
  - `}`
  - `PolygonP* PgAllocB(int n){`
    - `if(n < 0) return NULL;`
    - `int szHead = sizeof(short) + sizeof(Point*);`
    - `int szData = n * sizeof(Point);`
    - `void* buf = calloc(szHead + szData, 1);` =
    - `if(buf == NULL) return NULL;`
    - `PolygonP* pg = (Polygon*)buf;`
    - `pg->nVers = n;`
    - `pg->Vers = (Point*)((char*)buf + szHead);`
    - `return pg;`
  - `}`



cấp phát liên tục:  
Hủy, truy xuất nhanh; Hạn chế: RAM k đú, defragment

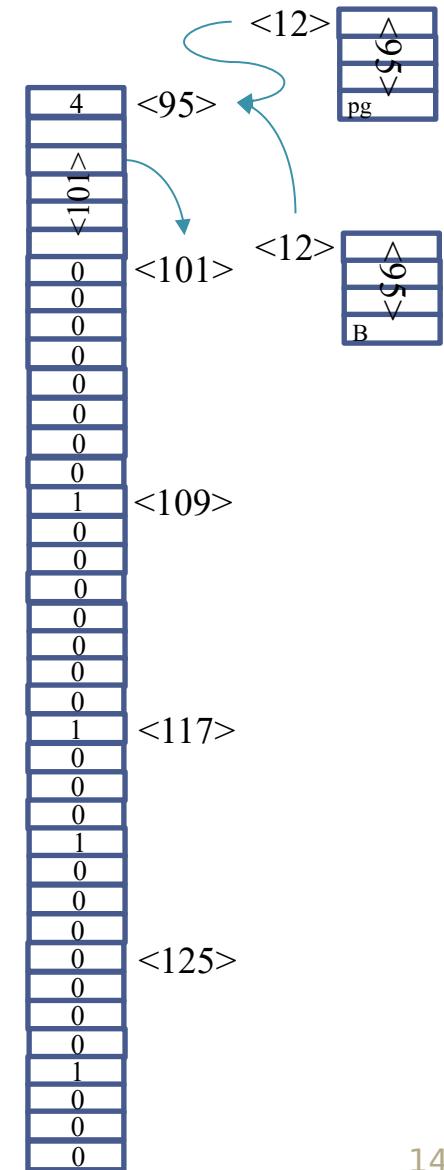
# DỮ LIỆU CÓ CẤU TRÚC

- Ví dụ với PolygonP (cách 2)
    - **typedef struct** {
      - long x, y;
    - } Point;
    - **typedef struct**{
      - short nVers; Point\* Vers;
    - } PolygonP;
    - **void main()**{
      - Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};
      - PolygonP\* B = PgAllocB(4);
      - **if**(B != NULL){
        - **for**(int i = 0; i < 4; i++){
          - B->Vers[i] = P[i];
        - }
      - }
    - }



# DỮ LIỆU CÓ CẤU TRÚC

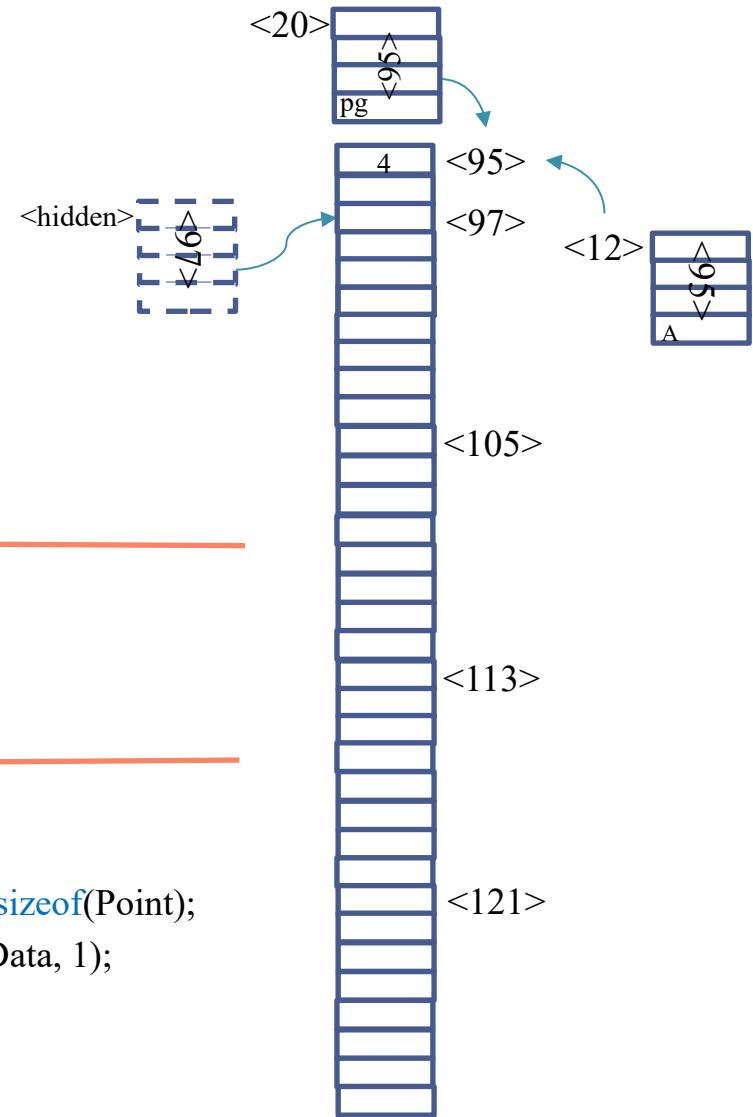
- Ví dụ với PolygonP (cách 2)
  - `typedef struct { long x, y; } Point;`
  - `typedef struct{`
    - `short nVers; Point* Vers;`
  - `} PolygonP;`
  - `void main(){`
    - `Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};`
    - `PolygonP* B = PgAllocB(4);`
    - `if(B != NULL){`
      - `for(int i = 0; i < 4; i++) B->Vers[i] = P[i];`
      - `pgFreeB(B);`
    - `}`
  - `}`
  - `void pgFreeB(PolygonP* pg){`
    - `if(pg != NULL) free(pg);`
  - `}`



# DỮ LIỆU CÓ CẤU TRÚC

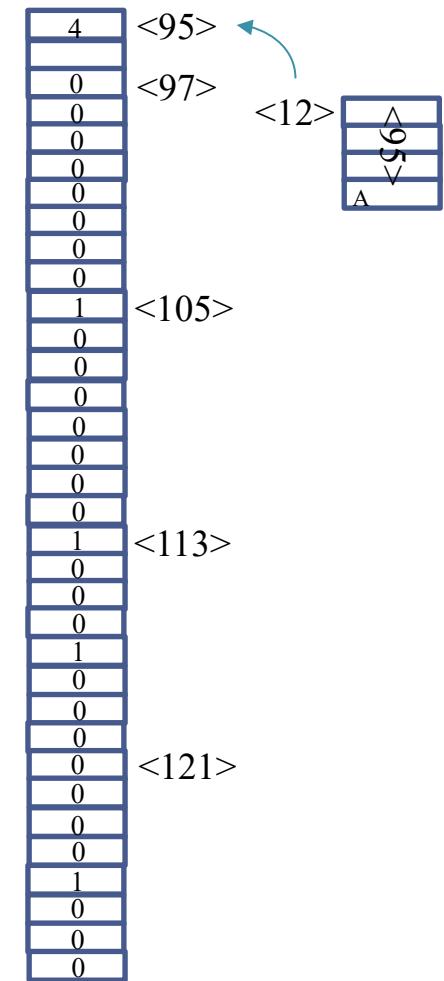
- Ví dụ với Polygon

- ```
typedef struct {
    long x, y;
} Point;
```
- ```
typedef struct{
    short nVers;
    Point Vers[1];
} Polygon;
```
- ```
void main(){
    Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};
    Polygon* A = PgAlloc(4);
}
```
- ```
Polygon* PgAlloc(int n){
    if(n < 0) return NULL;
    int szHead = sizeof(Polygon), szData = (n-1)*sizeof(Point);
    Polygon* pg = (Polygon*)calloc(szHead + szData, 1);
    if(pg == NULL) return NULL;
    pg->nVers = n;
    return pg;
}
```



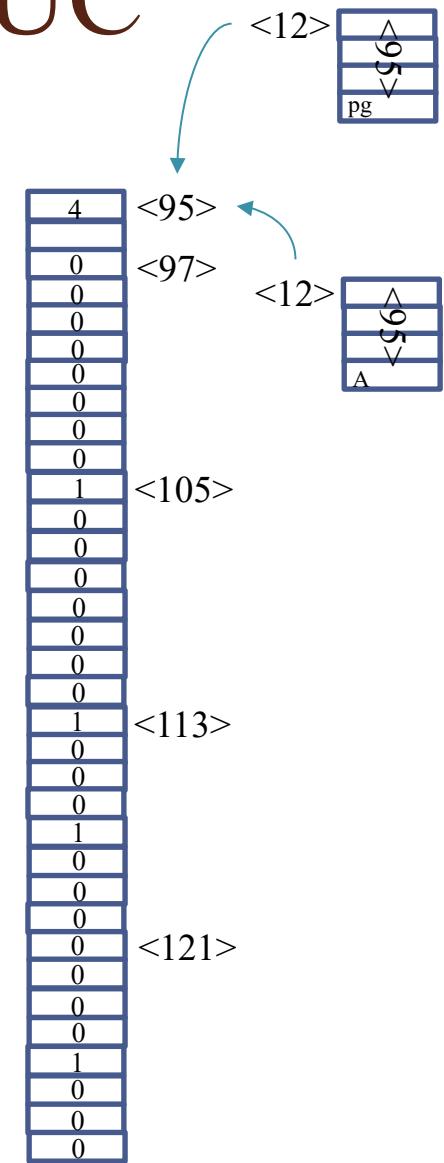
# DỮ LIỆU CÓ CẤU TRÚC

- Ví dụ với Polygon
  - **typedef struct** {
    - long x, y;
  - } Point;
  - **typedef struct**{
    - short nVers;
    - Point Vers[1];
  - } Polygon;
  - **void main()**{
    - Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};
    - Polygon\* A = PgAlloc(4);
    - **if**(A != NULL){
      - **for**(int i = 0; i < A->nVers; i++)
        - A->Vers[i] = P[i];
  - }



# DỮ LIỆU CÓ CẤU TRÚC

- Ví dụ với Polygon
  - `typedef struct { long x, y; } Point;`
  - `typedef struct{`
    - `short nVers;`
    - `Point Vers[1];`
  - `} Polygon;`
  - `void main(){`
    - `Point P[] = {{0, 0}, {1, 0}, {1, 1}, {0, 1}};`
    - `Polygon* A = PgAlloc(4);`
    - `if(A != NULL){`
    - `for(int i = 0; i < A->nVers; i++)`
      - `A->Vers[i] = P[i];`
    - `pgFree(A);`
  - `}`
  - `void pgFree(Polygon* pg){`
    - `if(pg != NULL) free(pg);`
  - `}`



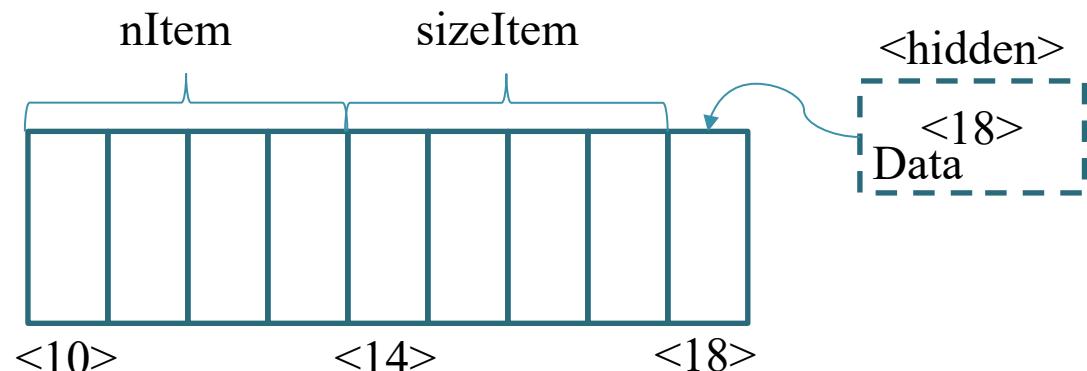
# CẤU TRÚC VÙNG NHỚ THÔ

- Dùng biến cấu trúc thay cho việc tính toán số học trên địa chỉ bộ nhớ
- Giúp mã nguồn trùu tượng, dễ đọc và dễ bảo trì
- Chương trình hiệu quả hơn do không cần tính toán số học trên địa chỉ bộ nhớ
- Áp dụng biến cấu trúc cho một số hàm trên mảng một chiều và mảng hai chiều

# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG MỘT CHIỀU)

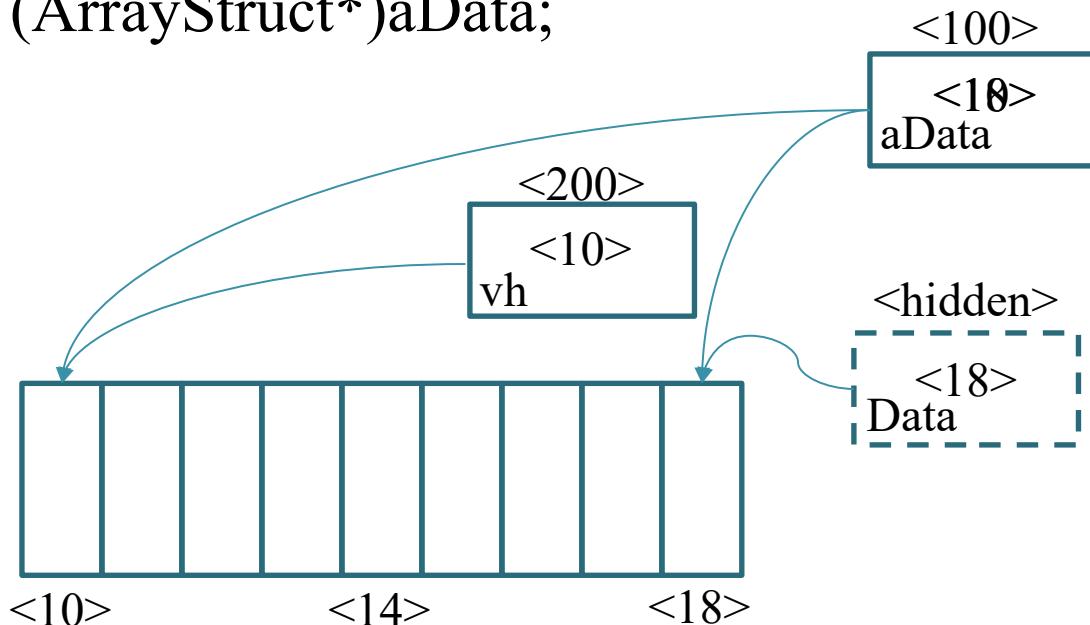
Don't forget: `pragma pack(1)`

- Xét lại ví dụ mảng một chiều
  - `typedef struct {`
  - `int nItem, sizeItem;`
  - `char Data[1];`
  - `}ArrayStruct;`
  - `static int headSize = sizeof(int) + sizeof(int);`



# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG MỘT CHIỀU)

- Xét lại ví dụ mảng một chiều
  - `ArrayStruct* StructOf(void* aData){`
    - `if(aData != NULL)`
      - `aData = (char*)aData - headSize;`
      - `return (ArrayStruct*)aData;`
  - `}`

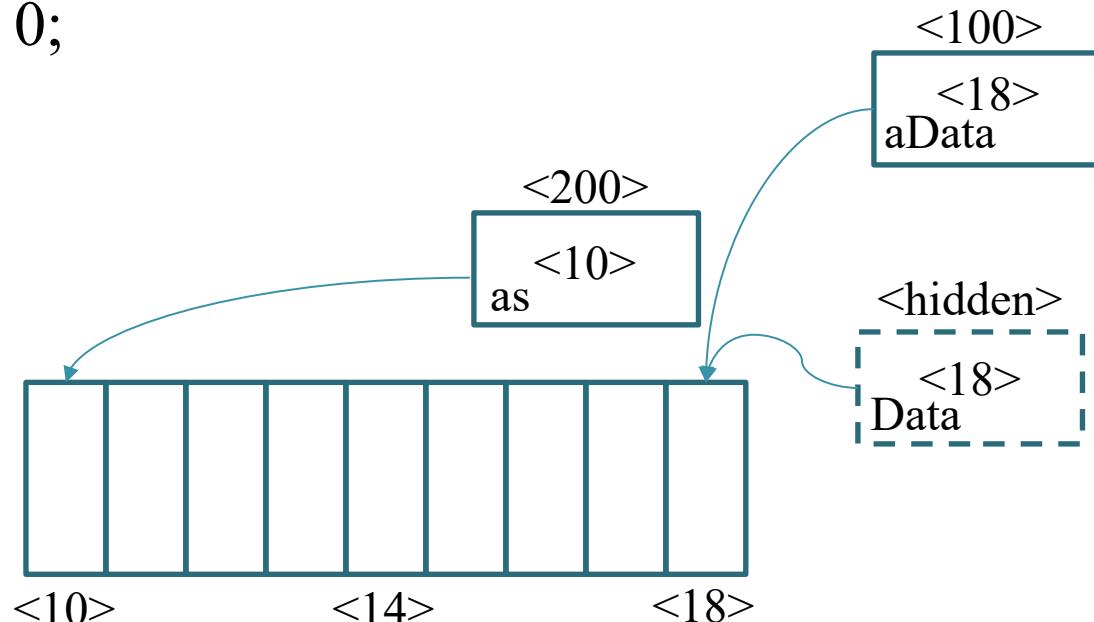


# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG MỘT CHIỀU)

- Xét lại ví dụ mảng một chiều

- `int arrSize(void* aData){`
    - `ArrayStruct* as = StructOf(aData);`
    - `if(as != NULL) return as->nItem;`
    - `return 0;`
  - `}`

return num of element  
in array  
4 first cells

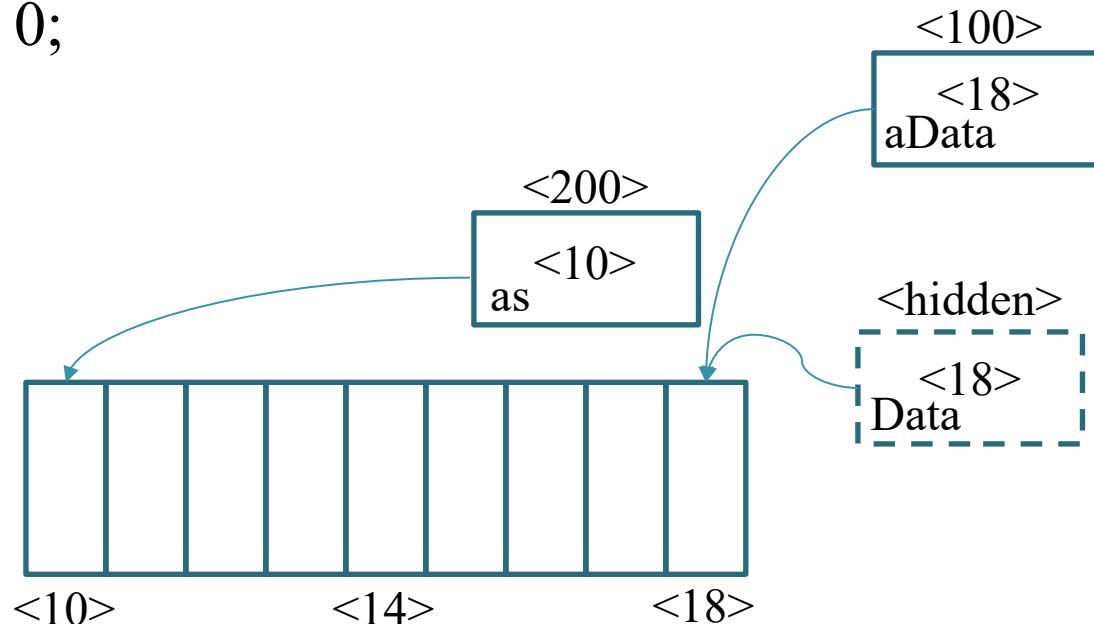


# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG MỘT CHIỀU)

- Xét lại ví dụ mảng một chiều

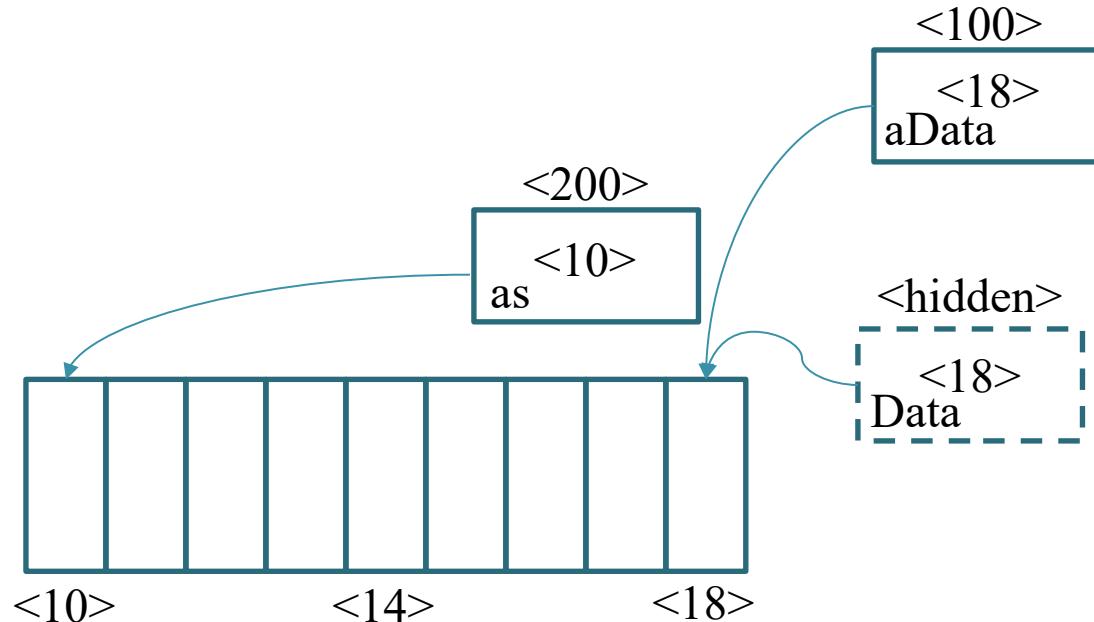
- `int arrItemSize(void* aData){`
    - `ArrayStruct* as = StructOf(aData);`
    - `if(as != NULL) return as->sizeItem;`
    - `return 0;`
  - `}`

return size of an element



# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG MỘT CHIỀU)

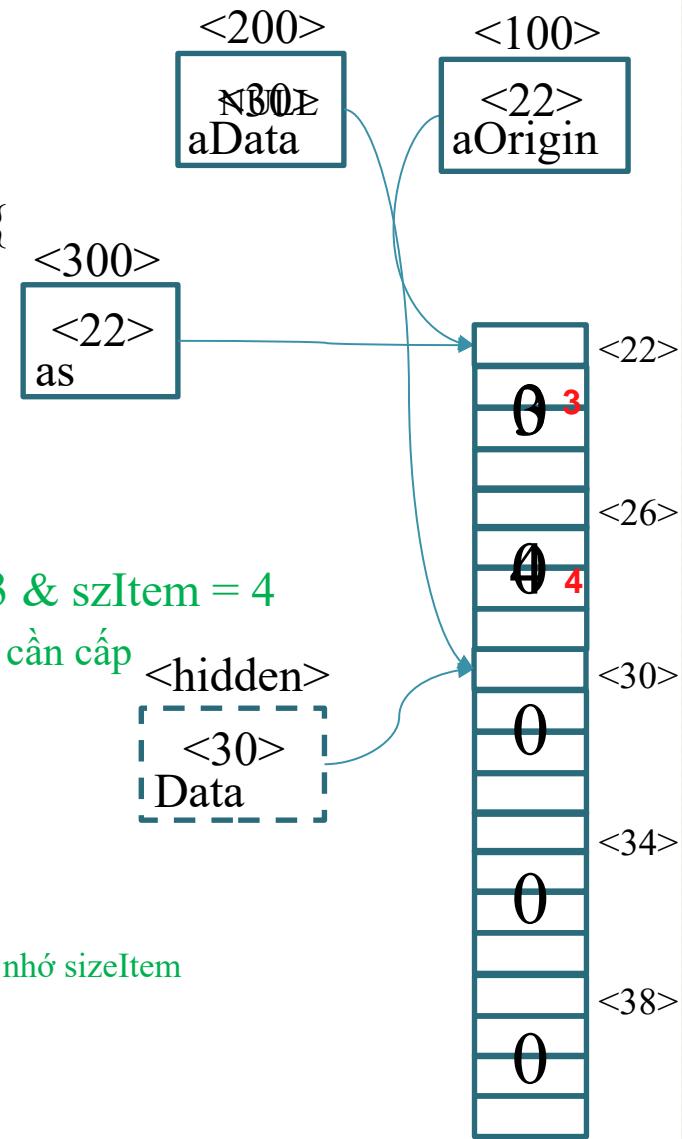
- Xét lại ví dụ mảng một chiều
  - `void arrFree(void* aData){`
    - `ArrayStruct* as = StructOf(aData);`
    - `if(as != NULL) free(as);`
  - `}`



# CẤU TRÚC VÙNG NHỚ THÔ (MẢNG MỘT CHIỀU)

- Xét lại ví dụ mảng một chiều
  - static int memSize(int nItem, int sizeItem){
    - if(sizeItem < 0) sizeItem = -sizeItem;
    - if(sizeItem == 0) sizeItem = 1;
    - if(nItem < 0) nItem = -nItem;
    - return headSize + nItem \* sizeItem;
  - } **tổng dung lượng cần thiết**
  - void\* arrInit(int n, int szItem){ //ví dụ n = 3 & szItem = 4
    - int sz = memSize(n, szItem); // Tính lượng byte cần cấp
    - void\* aOrigin = malloc(sz), \*aData = NULL;
    - if(aOrigin != NULL){
      - ArrayStruct\* as = (ArrayStruct\*)aOrigin;
      - memset(aOrigin, 0, sz);
      - as->nItem = n; // Gán giá trị nItem vào vùng nhớ nItem
      - as->sizeItem = szItem; // Gán giá trị sizeItem vào vùng nhớ sizeItem
      - aData = as->Data;
    - }
    - return aData;
  - }

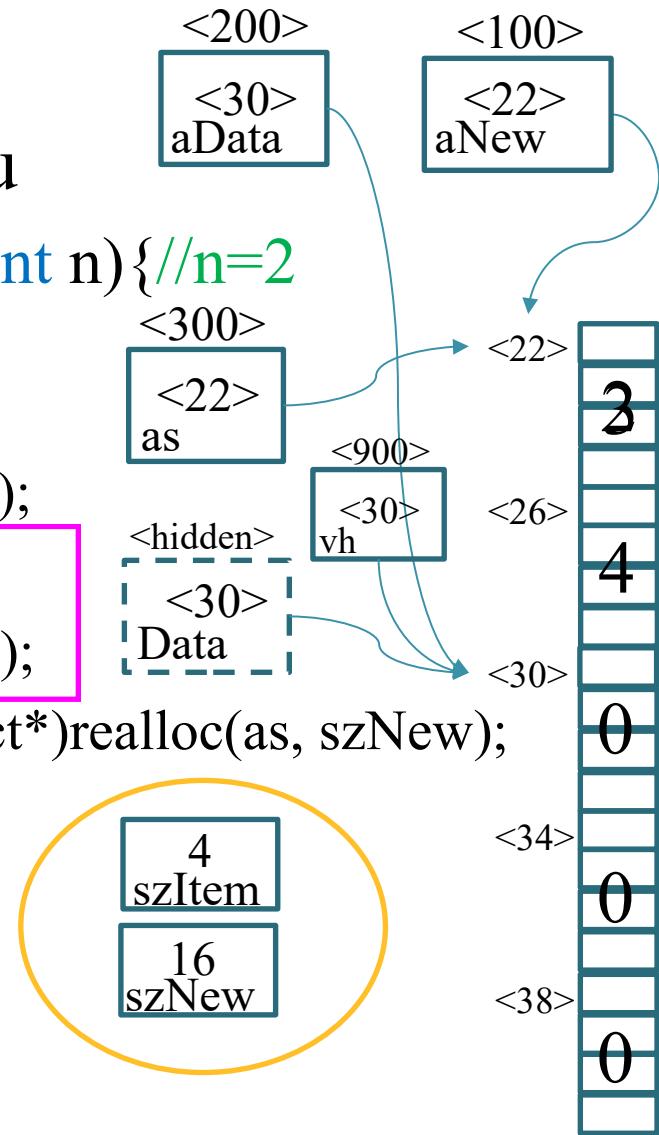
alloc2D ~ arrInit



# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG MỘT CHIỀU)

- Xét lại ví dụ mảng một chiều

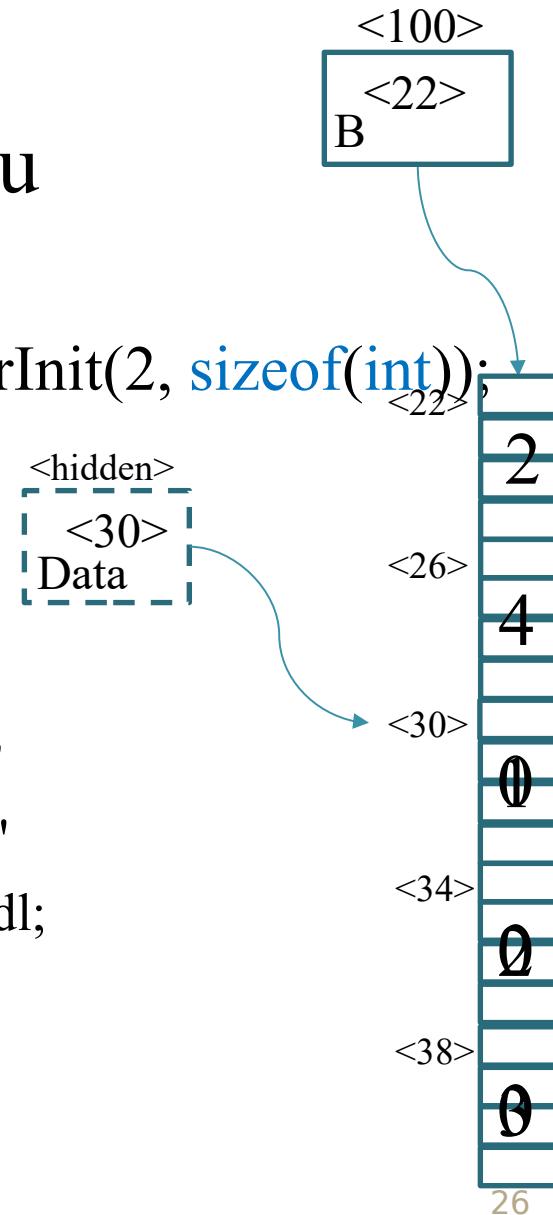
```
◦ void* arrResize(void* aData, int n){ //n=2
    • if(aData == NULL || n < 0)
        • return NULL;
    • ArrayStruct* as = StructOf(aData);
    • int szItem = as->sizeItem;
    • int szNew = memSize(n, sizeItem); highlighted code block
    • ArrayStruct* aNew = (ArrayStruct*)realloc(as, szNew);
    • if(aNew != NULL){
        • aNew->nItem = n;      update 3 --> 2
        • return aNew->Data;
    • }
    • return NULL;
    • }
```



# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG MỘT CHIỀU)

- Xét lại ví dụ mảng một chiều

- `void main(){`
  - `ArrayStruct*B=(ArrayStruct*)arrInit(2, sizeof(int));`
  - `if(B != NULL){`
    - `*((int*)(B->Data) + 0) = 1;`
    - `*((int*)(B->Data) + 1) = 2;`
    - `*((int*)(B->Data) + 2) = 3;`
    - `cout<< *((int*)(B->Data) + 0) << ", "`
    - `<< *((int*)(B->Data) + 1) << ", "`
    - `<< *((int*)(B->Data) + 2) << endl;`
    - `arrFree(B);`
  - `}`
- `}`



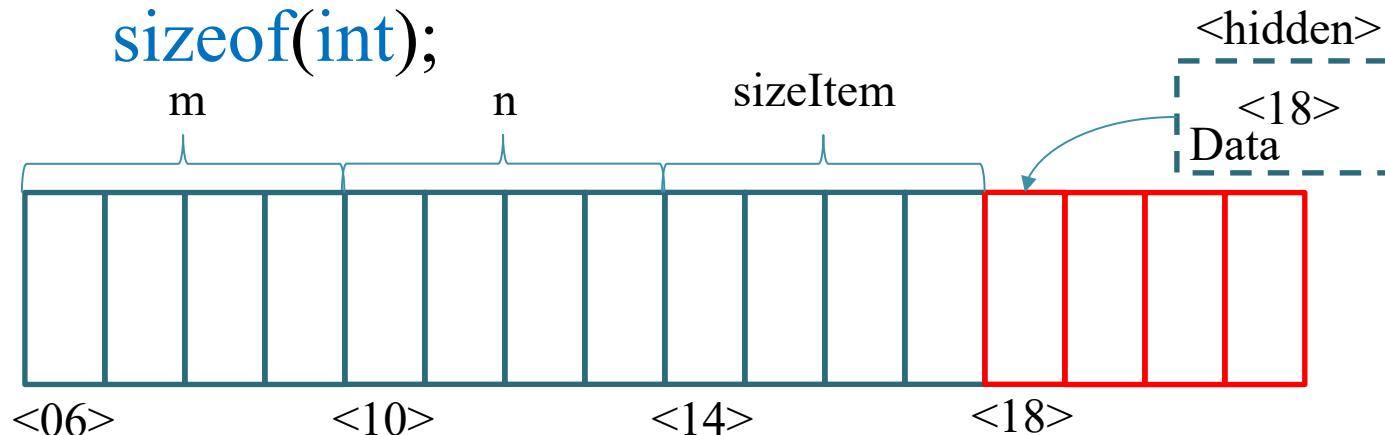
# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG HAI CHIỀU)

- Xét lại ví dụ mảng hai chiều

- `typedef struct {`
  - `int m, n, sizeItem;`
  - `void* Data[1];` **con trỏ dòng**

- `}aStruct;`

- `static int headSize = sizeof(int) + sizeof(int) +`  
`sizeof(int);`

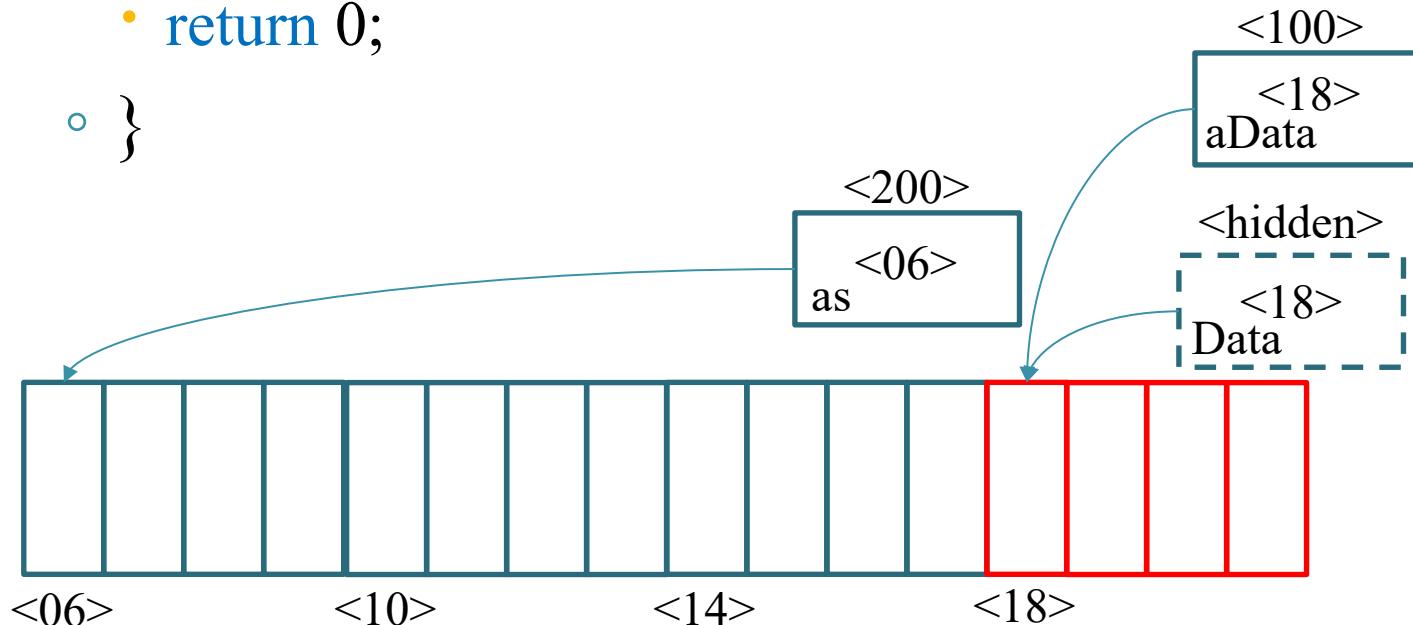


# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG HAI CHIỀU)

- Xét lại ví dụ mảng hai chiều

- ```
int nRow(void** aData){
```

  - aStruct\* as = (aStruct\*)((char\*)aData - headSize);
  - if(as != NULL) return as->m;
  - return 0;
- }

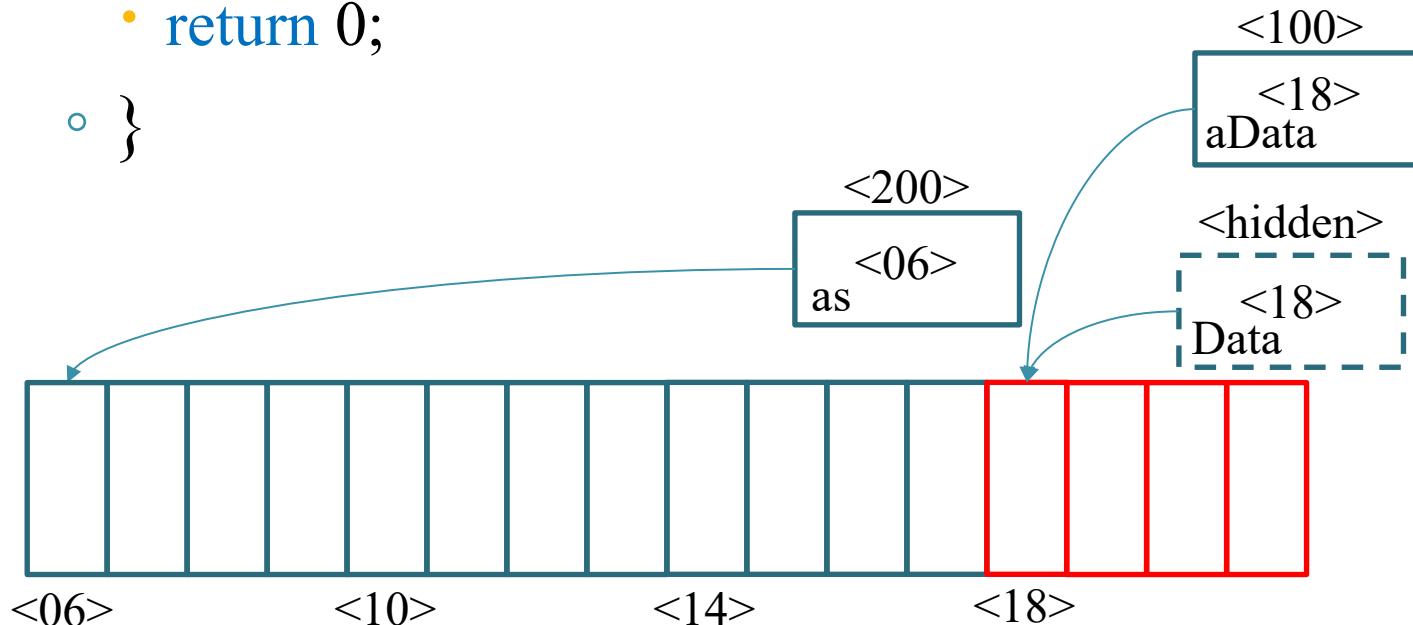


# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG HAI CHIỀU)

- Xét lại ví dụ mảng hai chiều

- ```
int nCol(void** aData){
```

  - aStruct\* as = (aStruct\*)((char\*)aData - headSize);
  - if(as != NULL) return as->n;
  - return 0;
- }



# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG HAI CHIỀU)

- Xét lại ví dụ mảng hai chiều

- `void Free2D(void** aData){`

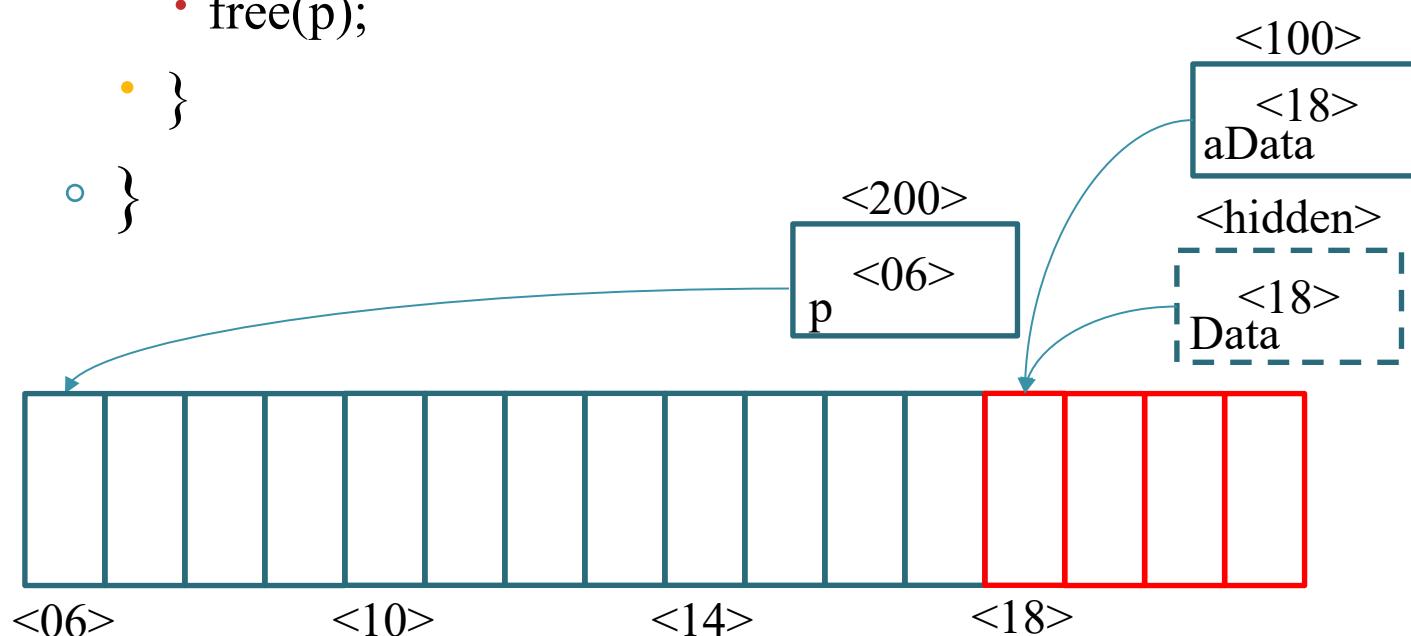
- `if(aData != NULL){`

- `void* p = (char*)aData - headSize;`

- `free(p);`

- }

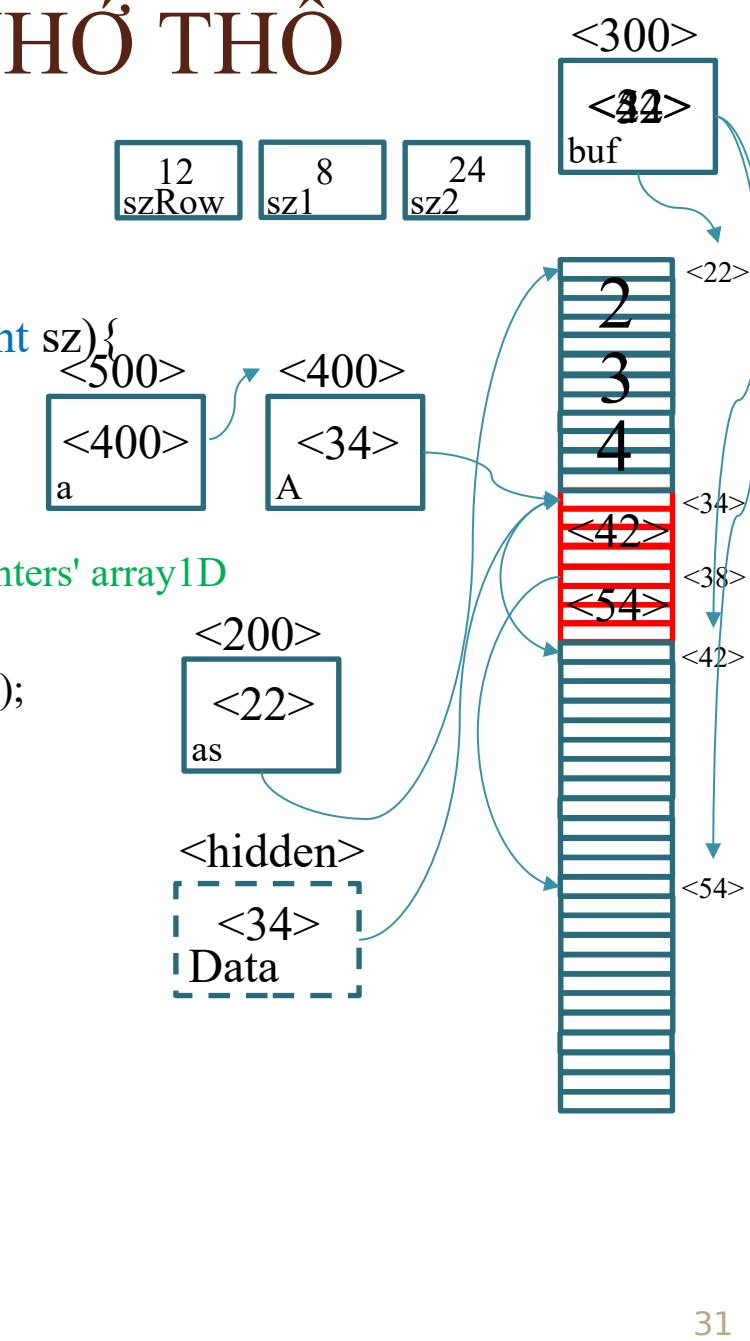
- }



# CẤU TRÚC VÙNG NHỎ THÔ (MẢNG HAI CHIỀU)

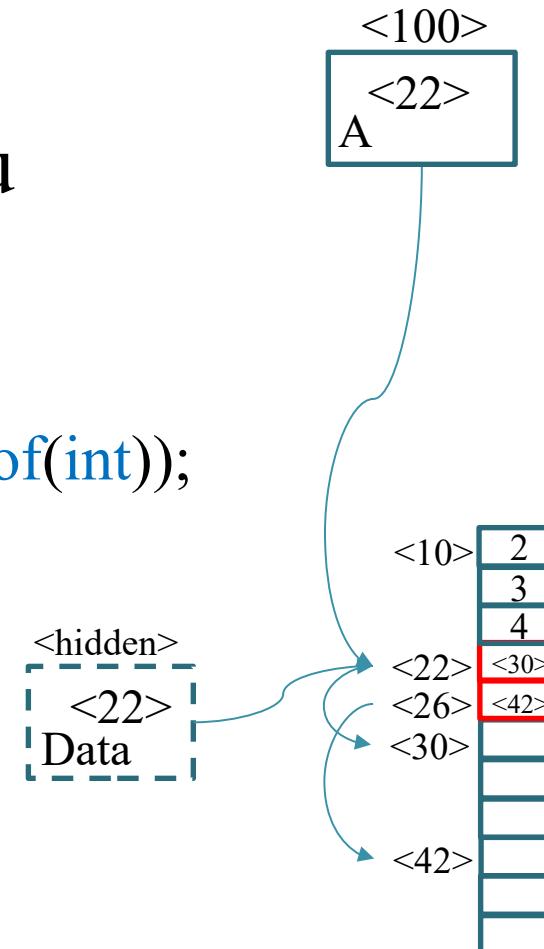
- Xét lại ví dụ mảng hai chiều

```
void alloc2D(void*** a, int m, int n, int sz){  
    //m = 2, n = 3, sz = 4  
    if(m <= 0 || n <= 0 || sz <= 0) return;  
    int szRow = n * sz; // bytes per row  
    int sz1 = m * sizeof(void*); // bytes of pointers' array1D  
    int sz2 = m * szRow; // bytes of all data  
    void* buf = calloc(headSize + sz1 + sz2, 1);  
    if(buf == NULL) return;  
    aStruct* as = (aStruct*)buf;  
    as->m = m; as->n = n; as->sizeItem = sz;  
    buf = (char*)buf + headSize + sz1;  
    as->Data[0] = buf;  
    for(int i = 1; i < m; i++){  
        buf = (char*)buf + szRow;  
        as->Data[i] = buf;  
    }  
    *a = as->Data;  
}
```



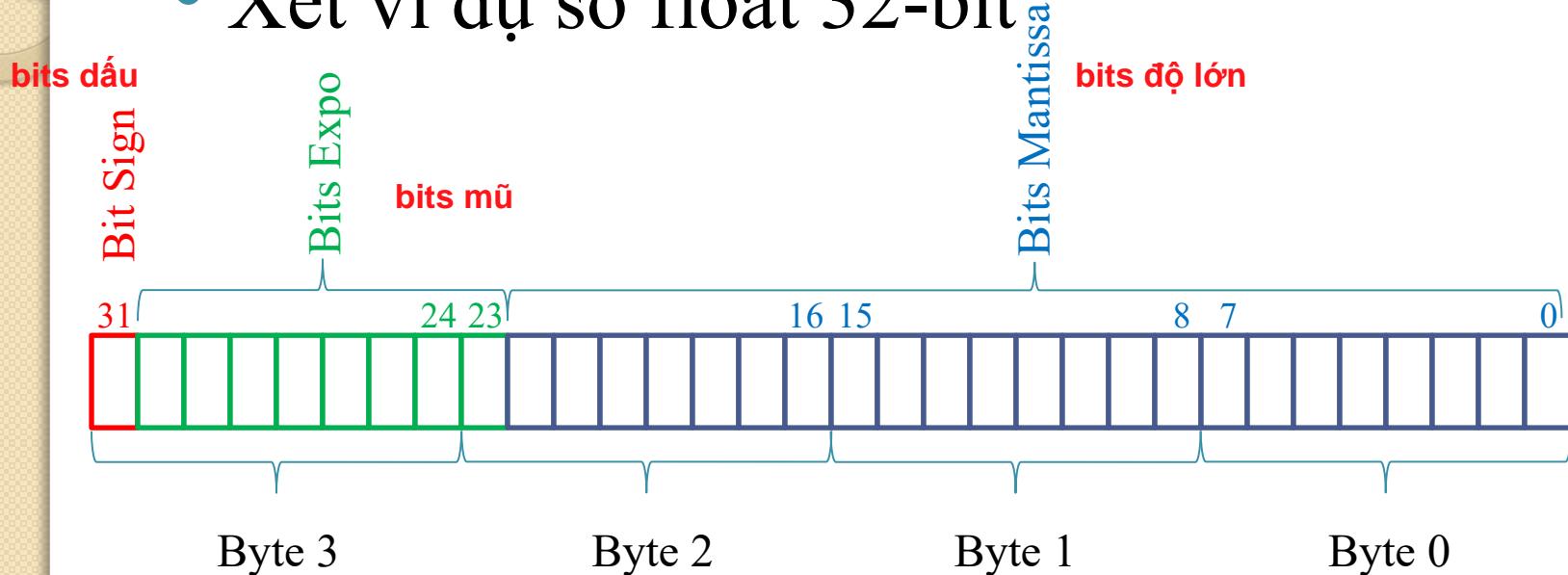
# CẤU TRÚC VÙNG NHỚ THÔ (MẢNG HAI CHIỀU)

- Xét lại ví dụ mảng hai chiều
  - `void main(){`
  - `int** A;`
  - `alloc2D((void***)&A, 2, 3, sizeof(int));`
  - `arr2D_Input(A);`
  - `arr2D_Output(A);`
  - `free2D((void**)A);`
  - `}`



# CẤU TRÚC VÙNG NHỎ THÔ (SỐ FLOAT CHUẨN 32-bit IEEE)

- Xét ví dụ số float 32-bit

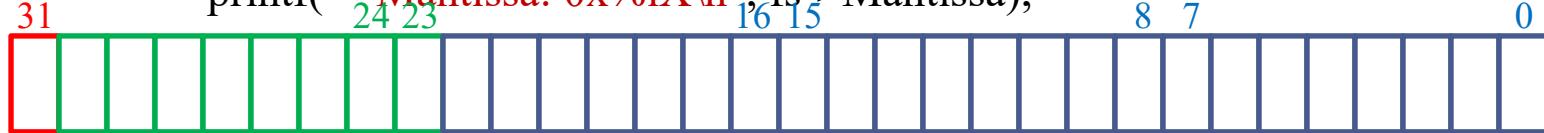


```
#pragma GCC diagnostic ignored "-Wpedantic"
typedef union{
    float Value; unsigned long dWord; unsigned short Words[2]; unsigned char Bytes[4];
    struct{ unsigned long Mantissa: 23; unsigned int Expo: 8; unsigned int Sign: 1; };
}floatStruct;      chọn kiểu maximum
```

# CẤU TRÚC VÙNG NHỚ THÔ (SỐ FLOAT CHUẨN 32-bit IEEE)

- Xét ví dụ số float 32-bit

```
void floatDump(floatStruct* fs){  
    if(fs == NULL) return;  
    printf("-----\n");  
    printf(" +value: %f\n", fs->Value);  
    printf(" +stored dword: 0x%lX\n", fs->dWord);  
    printf(" +stored words: 0x%04X 0x%04X\n", fs->Words[0], fs->Words[1]);  
    printf(" +stored bytes: ");  
    for(int i = 0; i < sizeof(float); i++)  
        printf("0x%02X ", fs->Bytes[i]);  
    printf("\n");  
    printf(" + IEEE stored parts:\n");  
    printf("   Sign: %d\n", fs->Sign);  
    printf("   Expo: 0x%X\n", fs->Expo);  
    printf("   Mantissa: 0x%lX\n", fs->Mantissa);
```



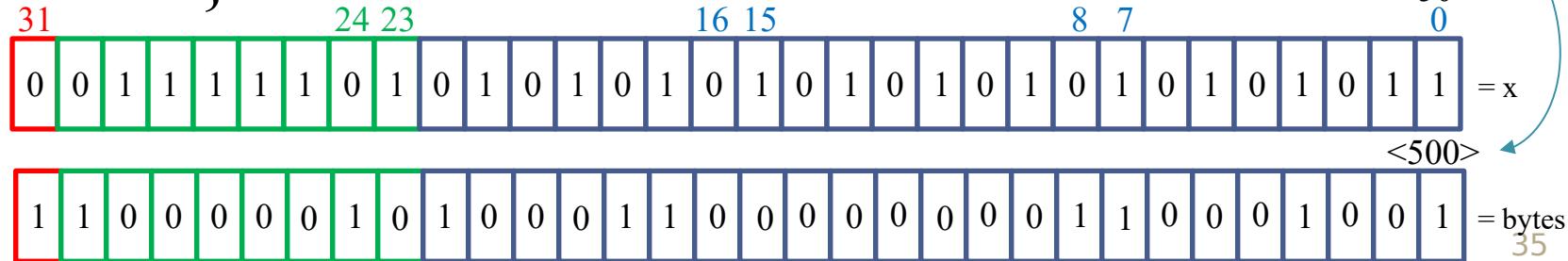
# CẤU TRÚC VÙNG NHỚ THÔ (SỐ FLOAT CHUẨN 32-bit IEEE)

- Xét ví dụ số float 32-bit

- void main(){

- float x = 1/(float)3;
    - unsigned char bytes[] = {0x89, 0x01, 0x46, 0xC1};
    - //bytes = -12.375375f
    - floatStruct\* p = (floatStruct\*)&x;
    - floatDump(p); p = (floatStruct\*)bytes;
    - floatDump(p);

- }



# CẤU TRÚC VÙNG NHỚ THÔ (SỐ FLOAT CHUẨN 32-bit IEEE)

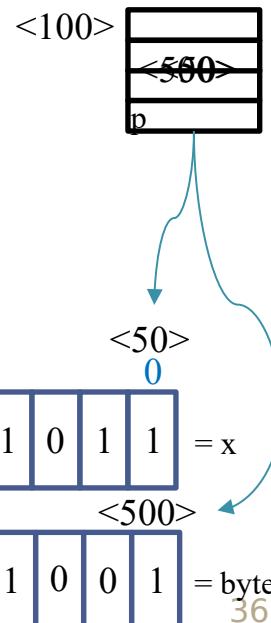
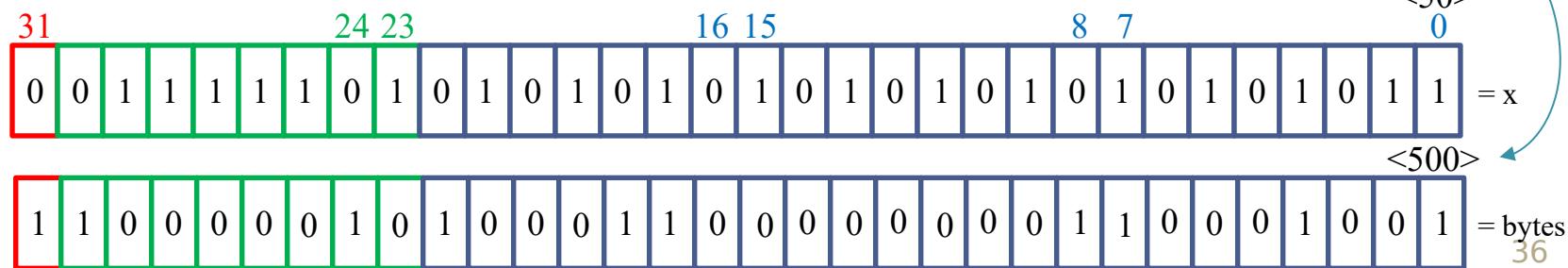
64bit?

- Xét ví dụ số float 32-bit

-----  
+ value: 0.333333  
+ stored dword: 0x3EAAAAAB  
+ stored words: 0xAAAB 0x3EAA  
+ stored bytes: 0xAB 0xAA 0xAA 0x3E  
+ IEEE stored parts:  
    Sign: 0  
    Expo: 0x7D  
    Mantissa: 0x2AAAAAB

1/3

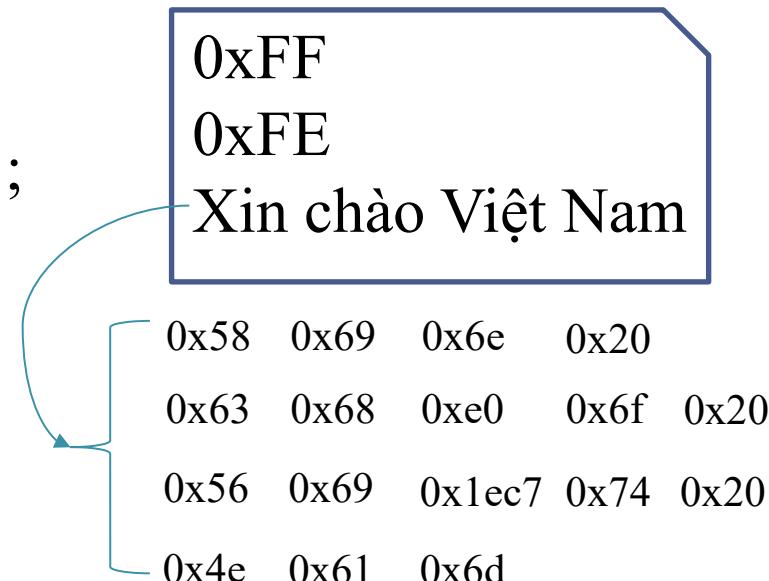
-----  
+ value: -12.375375  
+ stored dword: 0xC1460189  
+ stored words: 0x0189 0xC146  
+ stored bytes: 0x89 0x01 0x46 0xC1  
+ IEEE stored parts:  
    Sign: 1  
    Expo: 0x82  
    Mantissa: 0x460189



# CẤU TRÚC VÙNG NHỎ THÔ (TẬP TIN VĂN BẢN 16-BIT)

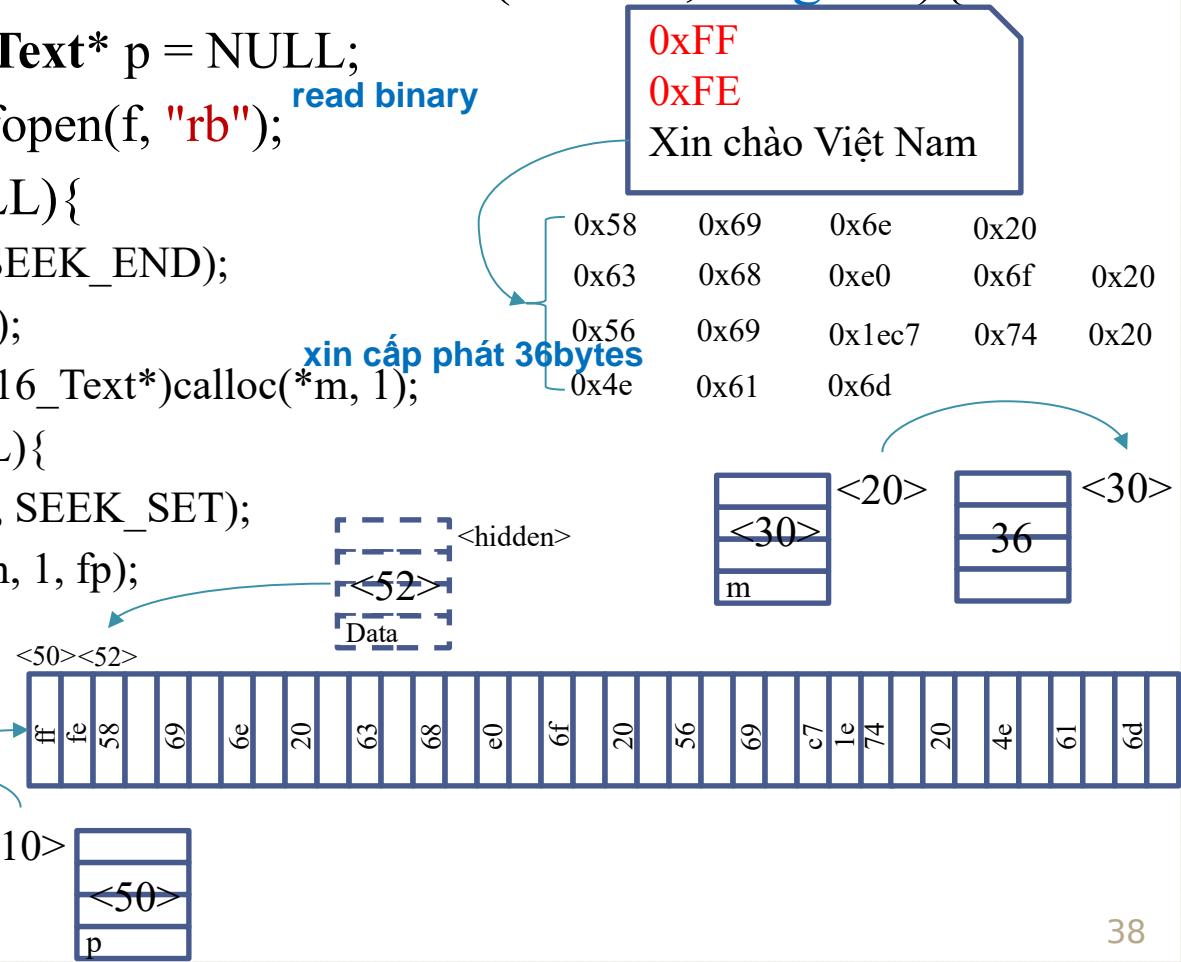
- Xét ví dụ tập tin văn bản 16-bit

```
typedef struct{
    union{
        unsigned char markBytes[2];
        unsigned short markWord;
    };
    1 ký tự chứa 2 bytes
    wchar_t Data[1];
}unicode16_Text;
```



# CẤU TRÚC VÙNG NHỎ THÔ (TẬP TIN VĂN BẢN 16-BIT)

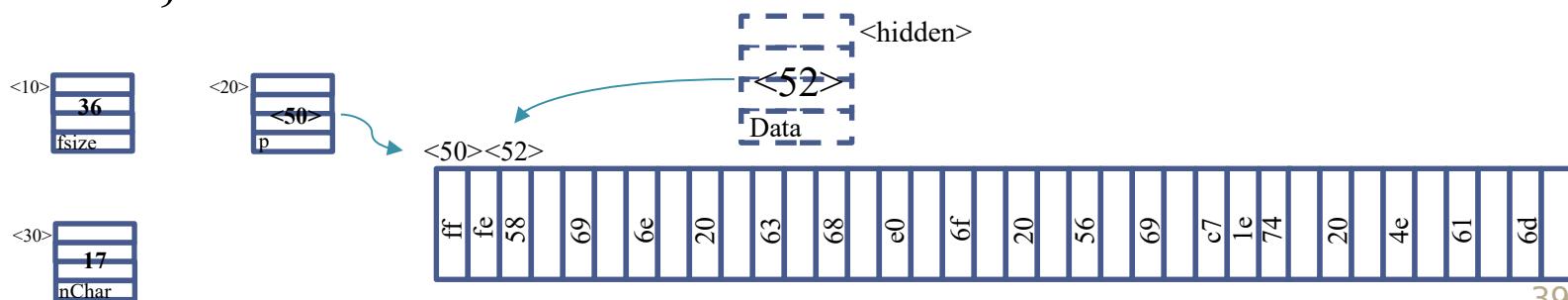
- Xét ví dụ tập tin văn bản 16-bit
  - **unicode16\_Text\*** readFileUtf16(**char\*** f, **long\*** m){
    - **unicode16\_Text\*** p = NULL;
    - FILE\* fp = fopen(f, "rb"); **read binary**
    - if(fp != NULL){
      - fseek(fp, 0, SEEK\_END);
      - \*m = ftell(fp);
      - ép kiểu **xin cấp phát 36bytes**
        - p = (**unicode16\_Text\***)calloc(\*m, 1);
        - if(p != NULL){
          - fseek(fp, 0, SEEK\_SET);
          - fread(p, \*m, 1, fp);
          - }
          - fclose(fp);
          - }
          - return p; **<10>**
    - }



# CẤU TRÚC VÙNG NHỎ THÔ (TẬP TIN VĂN BẢN 16-BIT)

- Xét ví dụ tập tin văn bản 16-bit

- void main(){
  - long fsize;
  - unicode16\_Text\* p = readFileUtf16("abc.inp", &fsize);
  - long nChar = (fsize - 2)/2;
  - if(p != NULL) cout << nChar << endl;
  - for(int i = 0; i < nChar; i++)
    - cout << hex << "0x" << (short)p->Data[i] << "|";
  - free(p);
- }



# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Toán tử chỉ số cho phép định nghĩa thêm ngoài các mảng kiểu cơ sở
- Áp dụng toán tử chỉ số có thể tiết kiệm vùng nhớ dành cho con trỏ (mảng 2 hay 3 chiều)
- Nguyên mẫu hàm của toán tử chỉ số cần trả về dạng tham chiếu
- Toán tử chỉ số nên định nghĩa trực tiếp bên trong khối lệnh ‘**struct**’

# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Ví dụ mảng vòng: xây dựng toán tử chỉ số để có thể tự do truyền chỉ số âm dương, hay vượt kích thước
- Ví dụ: a[-1] hay a[MAX + 1]...
  - template <class T>
  - struct roundArray{
    - int nItem;
    - T\* arr, Dummy;
    - //Cần '&' vì toán tử '[]' là left-exp của phép '='
    - T& operator[](int i){
      - if(arr == NULL || nItem <= 0) return Dummy;
      - if(i >= nItem) i = i % nItem;
      - else if(i < 0) {
        - int j = (-i) % nItem;
        - if(j == 0) i = 0;
        - else i = nItem - j;
      - }
      - return arr[i];
    - }

# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Ví dụ mảng vòng: xây dựng toán tử chỉ số để có thể tự do truyền chỉ số âm dương, hay vượt kích thước
- Ví dụ: a[-1] hay a[MAX + 1]...
  - `template <class T>`
  - `void roundArrayInit(roundArray<T>& a, int n){`
    - `a.nItem = 0; a.arr = NULL;`
    - `if(n <= 0) return;`
    - `a.nItem = n; a.arr = new T[n];`
  - `}`
  - `template <class T>`
  - `void roundArrayFree(roundArray<T>& a){`
    - `if(a.arr != NULL){`
    - `delete []a.arr; a.arr = NULL;`
    - `}`
  - `}`

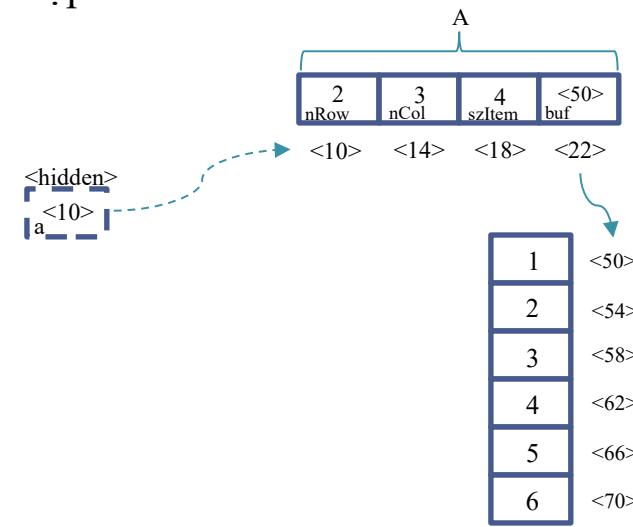
# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Ví dụ mảng vòng: xây dựng toán tử chỉ số để có thể tự do truyền chỉ số âm dương, hay vượt kích thước
- Ví dụ: a[-1] hay a[MAX + 1]...
  - `void main(){`
    - `roundArray<int> A; int n = 5;`
    - `roundArrayInit<int>(A, n);`
    - `for(int i = 0; i < A.nItem; i++) A[i] = i;`
    - `for(int i = -7; i <= 17; i++){`
      - `cout << " " << A[i];`
      - `if(i % n == 0) cout << endl;`
    - `}`
    - `roundArrayFree(A);`
  - `}`

# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Ví dụ mảng hai chiều: xây dựng cấu trúc hai chiều kết hợp toán tử chỉ số

```
• struct array2D{  
    •     int nRow, nCol, szItem; void* buf;  
    •     void* operator[](int i){  
        •         if(i < 0 || i >= nRow) i = 0;  
        •         return (char*)buf + i * nCol * szItem;  
    •     }  
    • };  
    • bool arr2D_Init(array2D& a, int m, int n, int sz){  
        •         if(m <= 0 || n <= 0 || sz <= 0) return false;  
        •         a.nRow = m; a.nCol = n; a.szItem = sz;  
        •         a.buf = calloc(m * n, sz);  
        •         if(a.buf == NULL) return false;  
        •         return true;  
    • }  
    • void arr2D_free(array2D& a){ if(a.buf != NULL) free(a.buf); }  
    • void main(){  
        •         array2D A;  
        •         if(arr2D_Init(A, 2, 3, sizeof(int))){  
            •             for(int i = 0; i < 2; i++)  
            •                 for(int j = 0; j < 3; j++)  
            •                     cin >> ((int*)A[i])[j]; // ví dụ nhập 1 2 3 4 5 6  
            •             arr2D_free(A);  
            •         }  
    • }
```



# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Ví dụ mảng 2 chiều: cấu trúc có template

		template <class T>
1	struct array2D{	struct array2D{
2	int nRow, nCol, szItem; void* buf;	int nRow, nCol; T* buf;
3	void* operator[](int i){	T* operator[](int i){
4	if(i < 0    i >= nRow) i = 0;	if(i < 0    i >= nRow) i = 0;
5	return (char*)buf + i * nCol * szItem;	return buf + i * nCol;
6	};	};
7		template <class T>
8	bool arr2D_Init(array2D& a, int m, int n, int sz){	bool arr2D_Init(array2D<T> &a, int m, int n){
9	if(m <= 0    n <= 0    sz <= 0) return false;	if(m <= 0    n <= 0) return false;
10	a.nRow = m; a.nCol = n; a.szItem = sz;	a.nRow = m; a.nCol = n;
11	a.buf = calloc(m * n, sz);	a.buf = (T*)calloc(m * n, sizeof(T));
12	if(a.buf == NULL) return false;	if(a.buf == NULL) return false;
13	return true;	return true;
14	}	}

# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Ví dụ mảng 2 chiều: cấu trúc có template

		template <class T>
15	void arr2D_free(array2D& a){	void arr2D_free(array2D<T> &a){
16	if(a.buf != NULL) free(a.buf);	if(a.buf != NULL) free(a.buf);
17	}	}
18	void main(){	void main(){
19	array2D A;	array2D<int> A;
20	if(arr2D_Init(A, 2, 3, sizeof(int))) {	if(arr2D_Init<int>(A, 2, 3)) {
21	for(int i = 0; i < 2; i++)	for(int i = 0; i < 2; i++)
22	for(int j = 0; j < 3; j++)	for(int j = 0; j < 3; j++)
23	cin >> ((int*)A[i])[j]; // ví dụ nhập 1 2 3 4 5 6	cin >> A[i][j]; // ví dụ nhập 1 2 3 4 5 6
24	arr2D_free(A);	arr2D_free<int>(A);
25	}	}

# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Ví dụ mảng ba chiều: xây dựng cấu trúc ba chiều kết hợp toán tử chỉ số

```


- template <class T>
- struct array3D{
    int nRow, nCol, nHigh; T* buf;
    array2D<T> operator[](int i){
        if(i < 0 || i >= nRow) i = 0;
        array2D<T> a2D; a2D.nRow = nCol; a2D.nCol = nHigh; a2D.buf = buf + i * nCol * nHigh;
        return a2D;
    };
}

```

```


- template <class T>
- bool arr3D_Init(array3D<T>& a, int m, int n, int r){

```

```


- if(m <= 0 || n <= 0 || r <= 0) return false;
    a.nRow = m; a.nCol = n; a.nHigh = r;
    a.buf = (T*)calloc(m * n * r, sizeof(T));
    if(a.buf == NULL) return false;
    return true;
}

```

```


- template <class T>

```

```


- void arr3D_free(array3D<T>& a){ if(a.buf != NULL) free(a.buf); }

```

```

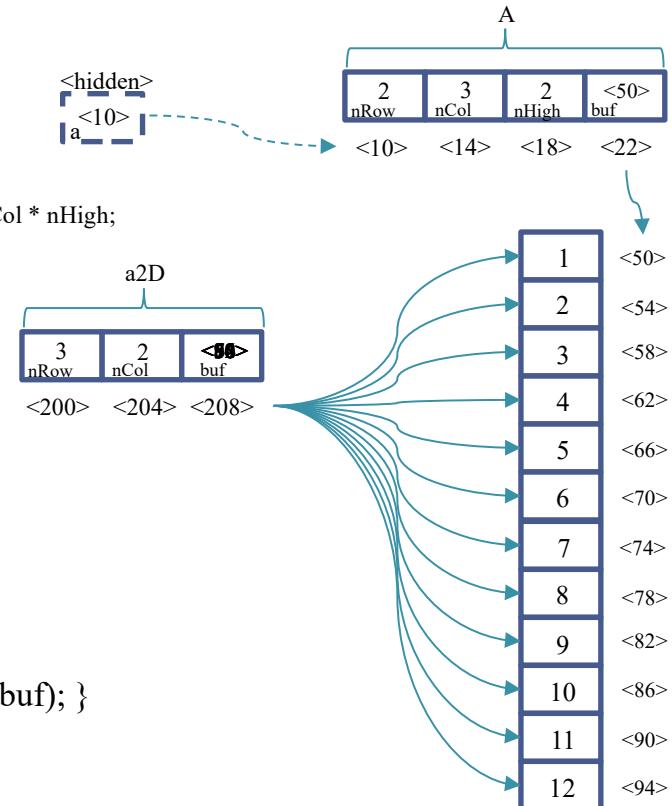

- void main(){
    array3D<int> A;
    if(arr3D_Init<int>(A, 2, 3, 2)){

```

```


- for(int i = 0; i < 2; i++)
        for(int j = 0; j < 3; j++)
            for(int k = 0; k < 2; k++)
                cin >> A[i][j][k]; // ví dụ nhập 1 2 3 4 5 6 7 8 9 10 11 12
    arr3D_free<int>(A);
}
}

```



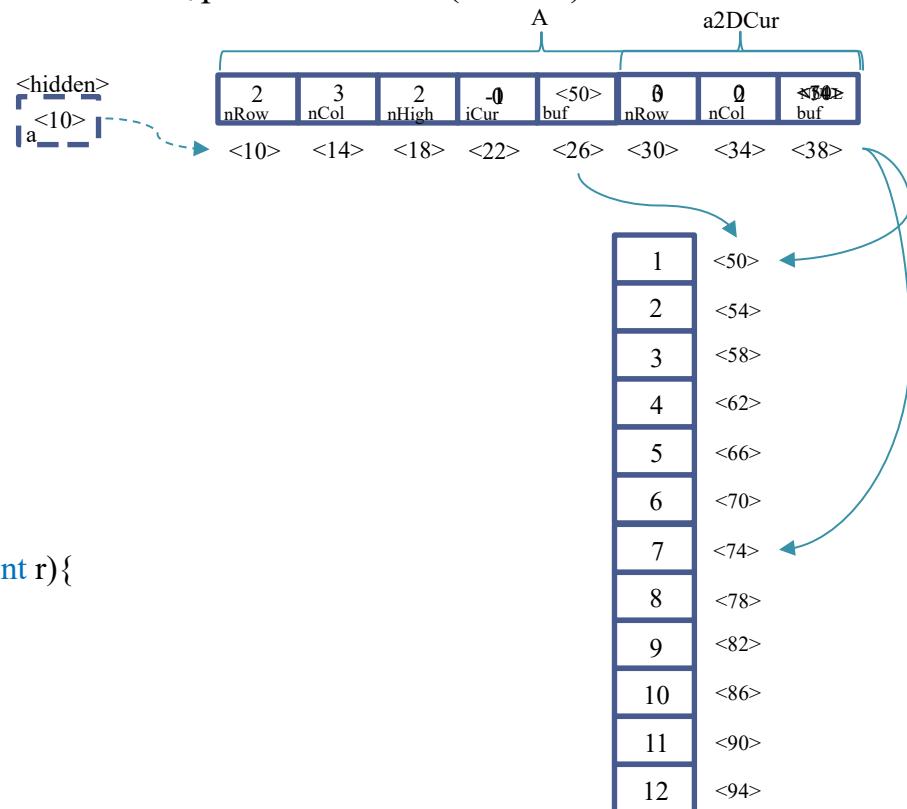
# TOÁN TỬ CHỈ SỐ BIẾN CẤU TRÚC

- Ví dụ mảng ba chiều: xây dựng cấu trúc ba chiều kết hợp toán tử chỉ số (cải tiến)

```

◦ template <class T>
◦ struct array3D{
    • int nRow, nCol, nHigh, iCur; T* buf;
    • array2D<T> a2DCur;
    • array2D<T> operator[](int i){
        • if(i < 0 || i >= nRow) i = 0;
        • if(i != iCur){
            • a2DCur.nRow = nCol; a2DCur.nCol = nHigh;
            • a2DCur.buf = buf + i * nCol * nHigh;
            • iCur = i;
        • }
        • return a2DCur;
    • };
    • template <class T>
    • bool arr3D_Init(array3D<T>& a, int m, int n, int r){
        • if(m <= 0 || n <= 0 || r <= 0) return false;
        • a.nRow = m; a.nCol = n; a.nHigh = r;
        • a.iCur = -1; a.a2DCur = {0};
        • a.buf = (T*)calloc(m * n * r, sizeof(T));
        • if(a.buf == NULL) return false;
        • return true;
    • }
    • template <class T>
    • void arr3D_free(array3D<T>& a){ //... }
    • void main(){
        • //...
    • }
}

```



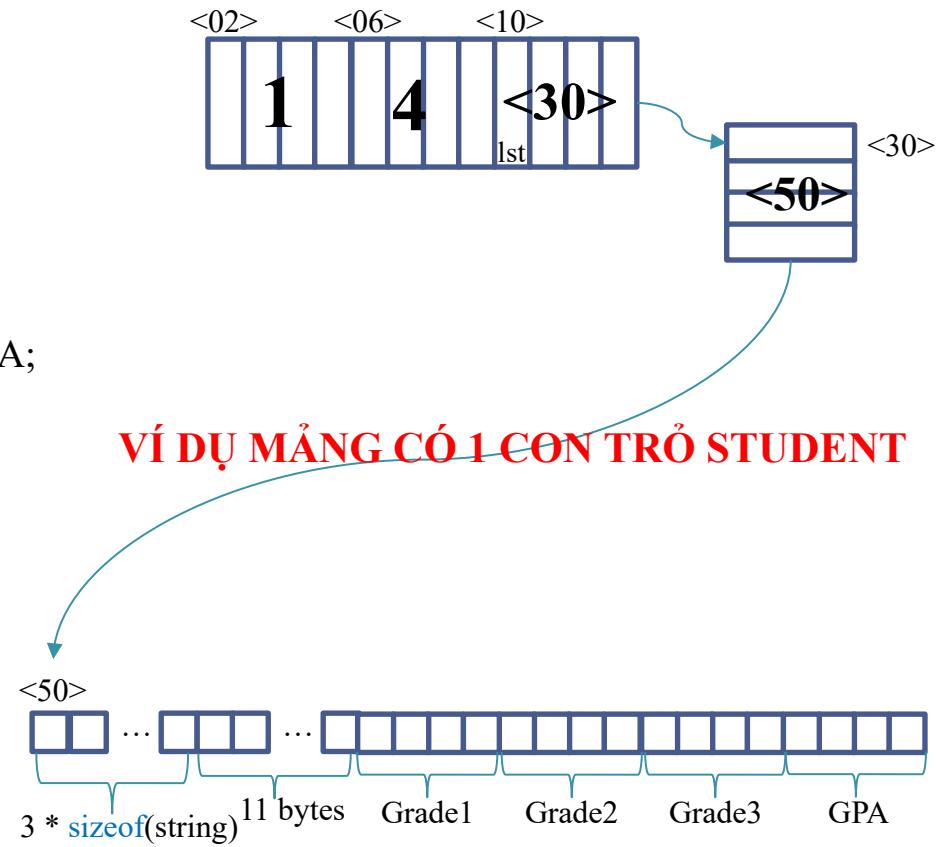
# MẢNG CẤU TRÚC

- Là mảng mà mỗi phần tử là một kiểu do người dùng tự định nghĩa (nhờ struct)
- Kiểu mới có thể có các trường cố định hoặc biến động
- Với trường cố định ta có thể cấp phát trực tiếp số lượng byte, có thể dùng các tạo mảng một chiều như các ví dụ trước
- Với trường biến động ta cần phải xử lý riêng

# MẢNG CẤU TRÚC

- Xét ví dụ cấu trúc Student

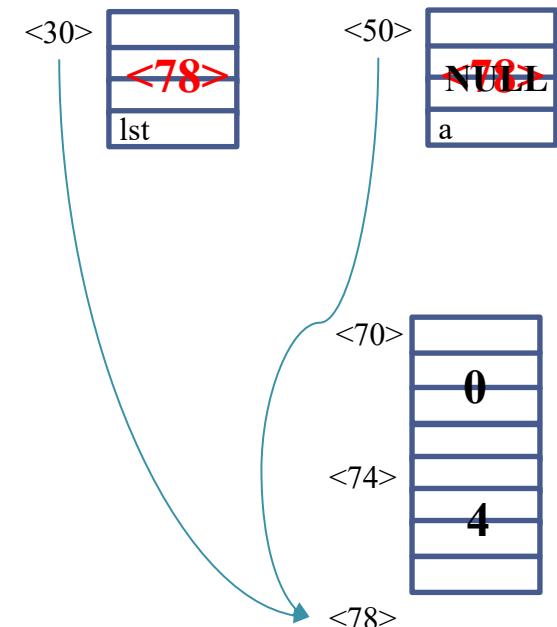
```
◦ typedef struct{  
    • string Code, FamilyName, Name;  
    • char BirthDate[11];  
    • float Grade1, Grade2, Grade3, GPA;  
◦ } Student;  
◦ void main(){  
    • Student** lst = stInit(0), st;  
    • string stcode = “nocode”;  
    • while(1){  
        • cin >> stcode;  
        • if(stcode == “---”) break;  
        • st.Code = stcode;  
        • getStudent(st);  
        • if(StPush(&lst, st) == 0) break;  
        • }  
    • int n = arrSize((void*)lst);  
    • for(int i = 0; i < n; i++) {//cout << lst[i]->Code ...  
    • StFree(lst);  
◦ }
```



# MẢNG CẤU TRÚC

- Xét ví dụ cấu trúc Student

- `typedef struct{`
  - `string Code, FamilyName, Name;`
  - `char BirthDate[11];`
  - `float Grade1, Grade2, Grade3, GPA;`
- `}` Student;
- `void main(){`
  - `Student** lst = stInit(0), st;`
  - `//...`
- `}`
- `Student** StInit(int n){`
  - `Student** a = NULL;`
  - `if(n < 0) n = 0;`
  - `a = (Student**)arrInit(n, sizeof(Student*));`
  - `return a;`
- `}`



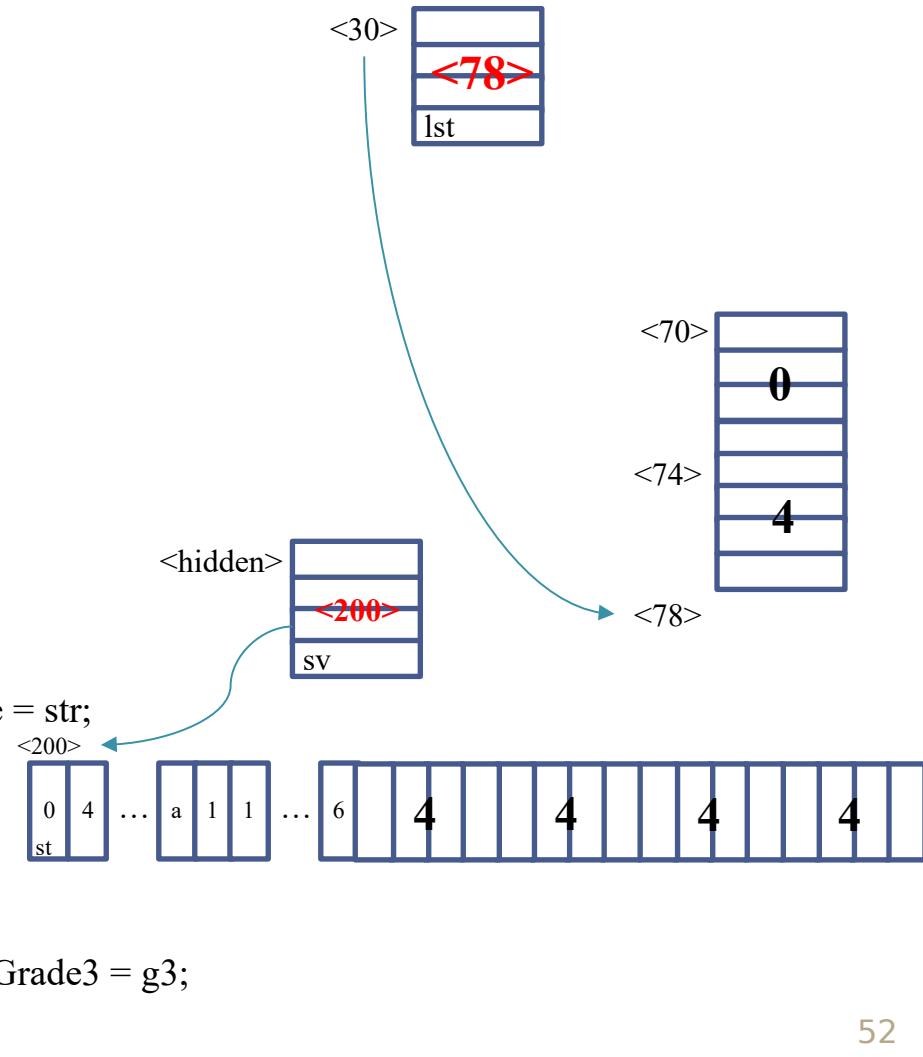
# MẢNG CẤU TRÚC

- Xét ví dụ cấu trúc Student

```

    void main(){
        Student** lst = stInit(0), st;
        string stcode = "nocode";
        while(1){
            cin >> stcode;
            if(stcode == "--") break;
            st.Code = stcode;
            getStudent(st);
            //...
        }
        void getStudent(Student& sv){
            cin.ignore();
            char str[256]; float g1, g2, g3;
            cin.getline(str, 256); sv.FamilyName = str;
            cin.getline(str, 256); sv.Name = str;
            cin.getline(sv.BirthDate, 256);
            cin >> g1 >> g2 >> g3;
            sv.GPA = (g1 + g2 + g3)/3;
            sv.Grade1 = g1; sv.Grade2 = g2; sv.Grade3 = g3;
        }
    }

```



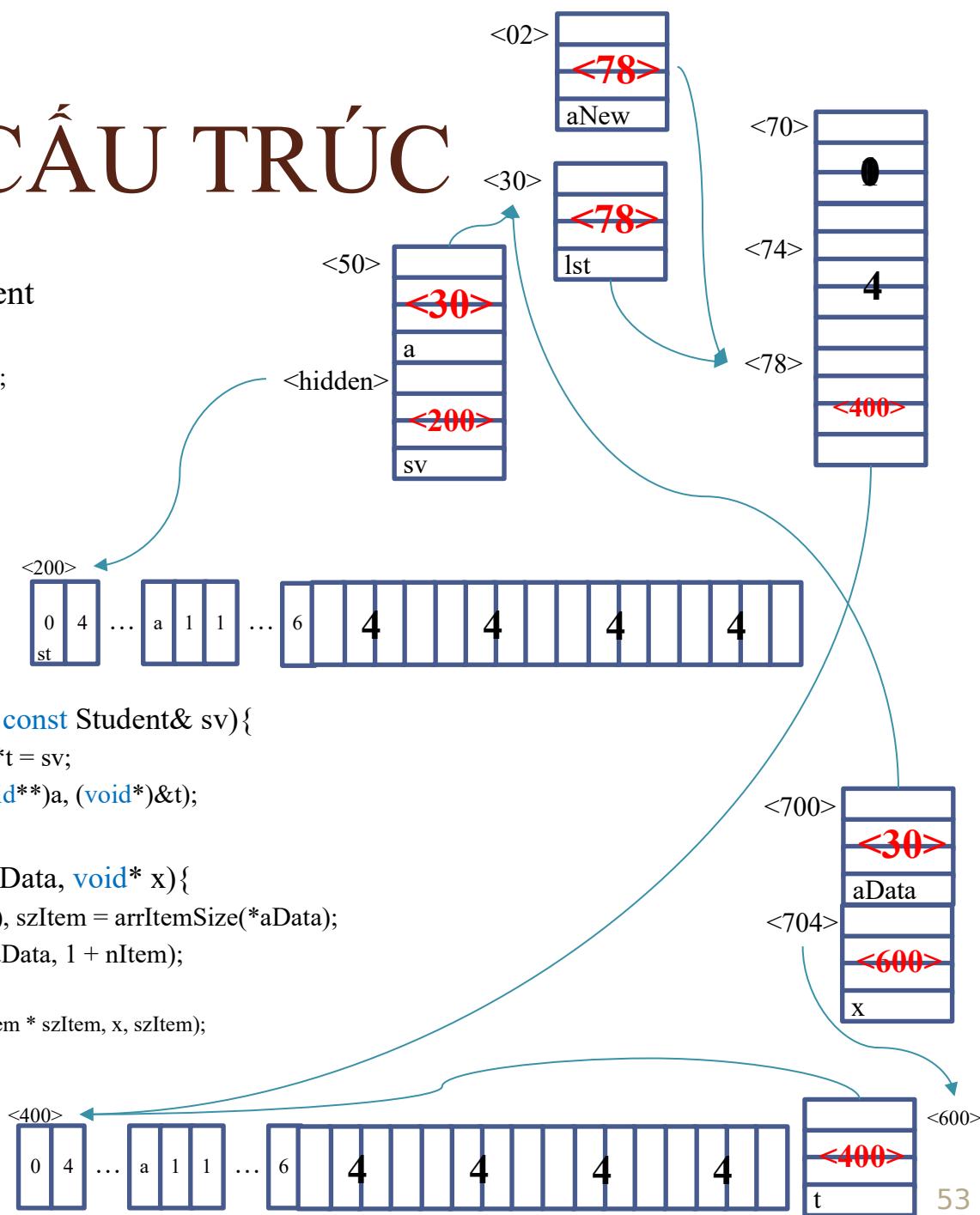
# MẢNG CẤU TRÚC

- Xét ví dụ cấu trúc Student

```

    void main(){
        Student** lst = stInit(0), st;
        string stcode = "nocode";
        while(1){
            cin >> stcode;
            if(stcode == "--") break;
            st.Code = stcode;
            getStudent(st);
            StPush(&lst, st);
            //...
        }
        int StPush(Student*** a, const Student& sv){
            Student* t = new Student; *t = sv;
            return arrPushback((void**)a, (void*)&t);
        }
        int arrPushback(void*** aData, void* x){
            int nItem = arrSize(*aData), szItem = arrItemSize(*aData);
            void* aNew = arrResize(*aData, 1 + nItem);
            if(aNew != NULL){
                memmove((char*)aNew + nItem * szItem, x, szItem);
                *aData = aNew;
                return 1;
            }
            return 0;
        }
    }

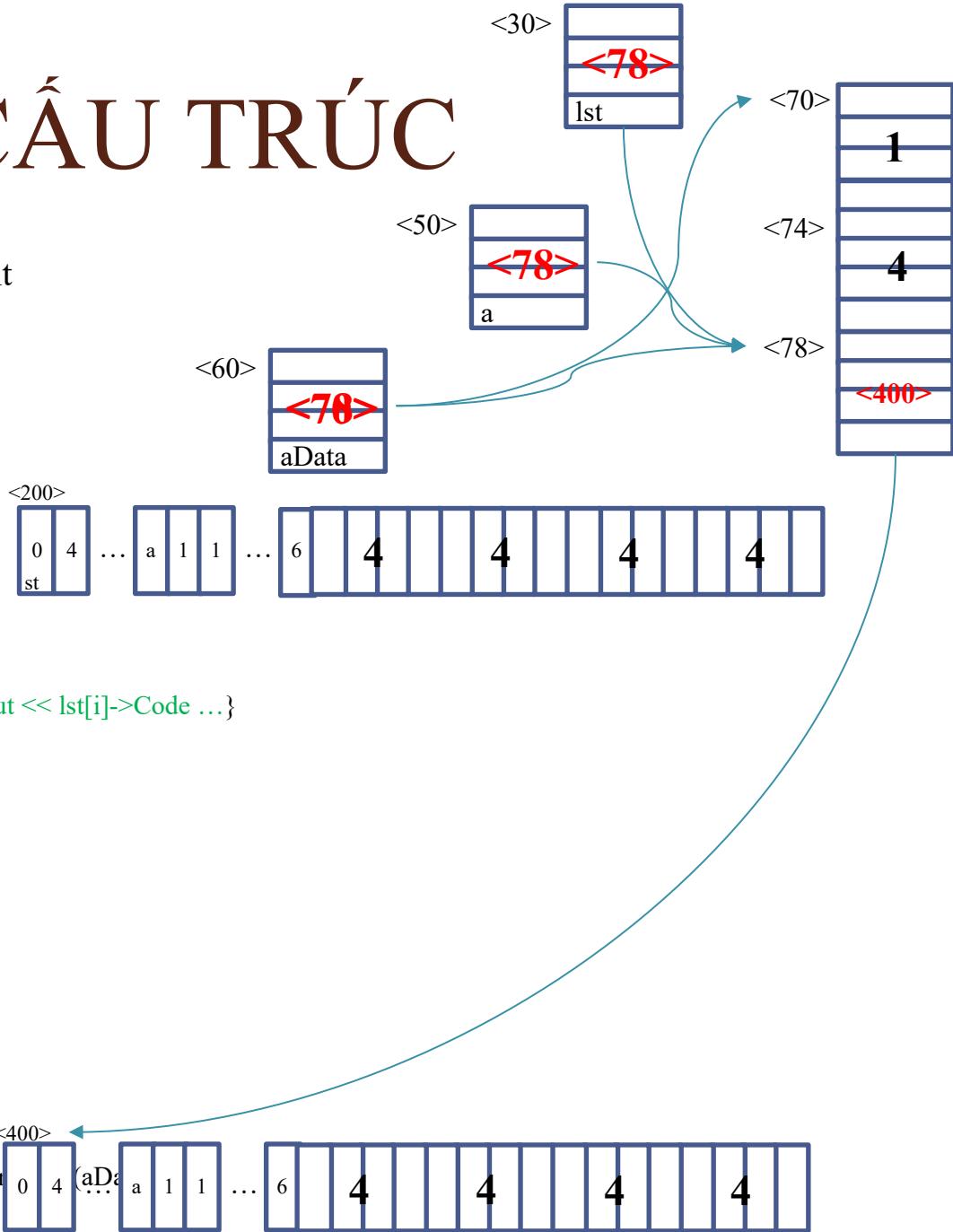
```



# MẢNG CẤU TRÚC

- Xét ví dụ cấu trúc Student

- **void main()**{
  - Student\*\* lst = **stInit(0)**, st;
  - string stcode = “**nocode**”;
  - **while**(1){
    - cin >> stcode;
    - if(stcode == “**...**”) **break**; <200>
    - st.Code = stcode;
    - **getStudent**(st);
    - **StPush**(&lst, st);
  - }
  - int n = **arrSize**((**void**\*)lst);
  - **for**(int i = 0; i < n; i++) {**//cout << lst[i]->C**
  - **StFree**(lst);
- }
- **void StFree(Student\*\* a)**{
  - if(a == NULL) **return**;
  - int n = **arrSize**(a);
  - **for**(int i = 0; i < n; i++){
    - if(a[i] != NULL) **delete** a[i];
  - }
  - **arrFree**(a);
- }
- **void arrFree(**void**\* aData)**{<400>
  - if(aData != NULL) **free**(origin a)
- }



# MẢNG CẤU TRÚC

- Xây dựng mảng cấu trúc các đa giác
- Sử dụng kỹ thuật đọc/ghi file nhị phân
- Tái sử dụng các hàm đã xây dựng trong Array1D
- Cấu trúc file đọc/ghi cơ bản như sau

Có m đa giác

$\{$

$<\text{số đỉnh } n_1><x_1><y_1> \dots <x_{n\_1}><y_{n\_1}>$

$<\text{số đỉnh } n_2><x_1><y_1> \dots <x_{n\_2}><y_{n\_2}>$

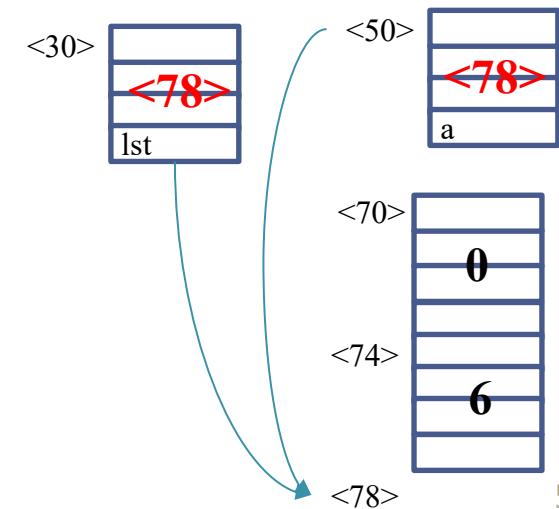
$\dots$

$<\text{số đỉnh } n_m><x_1><y_2> \dots <x_{n\_m}><y_{n\_m}>$

# MẢNG CẤU TRÚC

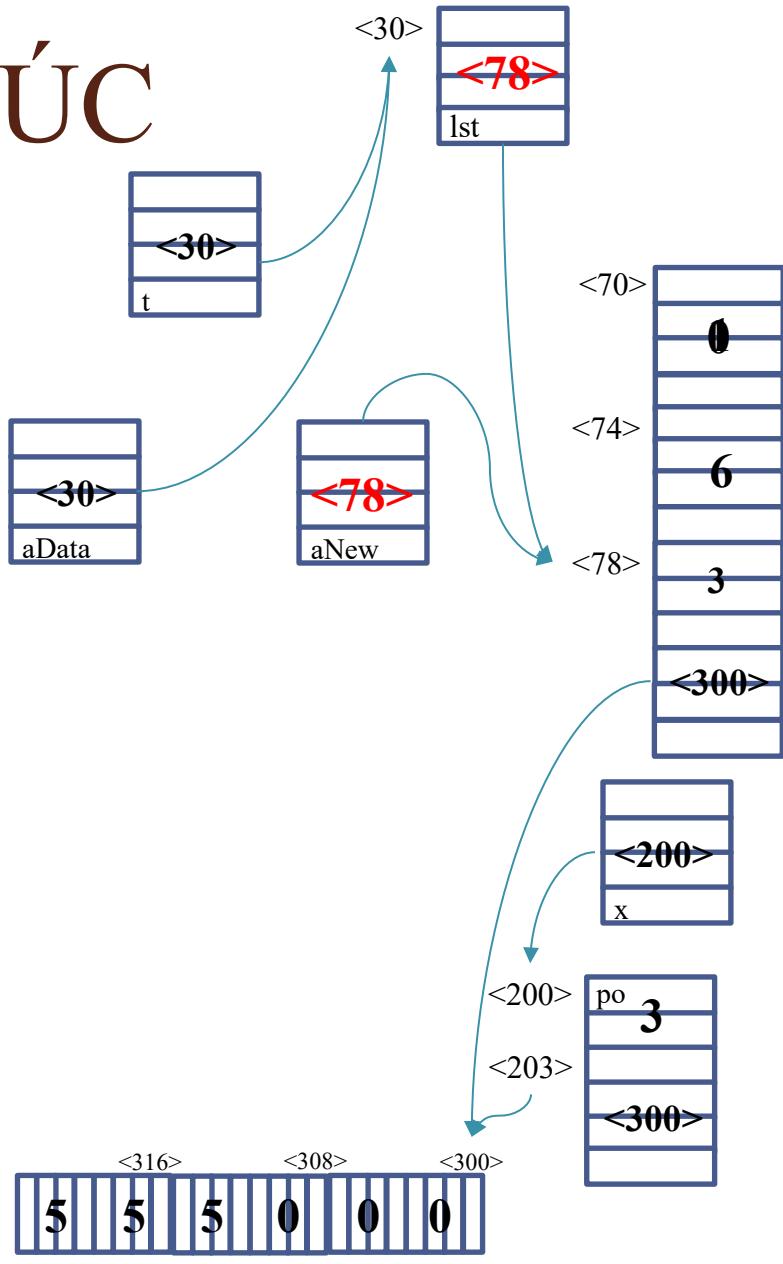
- Các khai báo struct cần thiết

- `typedef struct{ int x, y; } Point;`
- `typedef struct { short nVer; Point* Vers; } Polygon;`
- `void main(){`
  - `Point dg1[] = {{0, 0}, {0, 5}, {5, 5}}; int n1 = 3;`
  - `Point dg2[] = {{0, 0}, {0, 5}, {5, 5}, {5, 0}}; int n2 = 4;`
  - `Point dg3[] = {{0, 0}, {0, 5}, {1, 1}, {5, 5}, {5, 0}}; int n3 = 5;`
  - `Polygon* lst = PolyListInit();`
  - `if(lst != NULL){`
    - `PolyListPush(&lst, dg1, n1); PolyListPush(&lst, dg2, n2);`
    - `PolyListPush(&lst, dg3, n3);`
    - `PolyListSave(lst, "Polygons.dat");`
    - `PolyListFree(lst);`
  - `}`
- `}`
- `Polygon* PolyListInit(){`
  - `void* a = arrInit(0, sizeof(Polygon));`
  - `return (Polygon*)a;`
- `}`



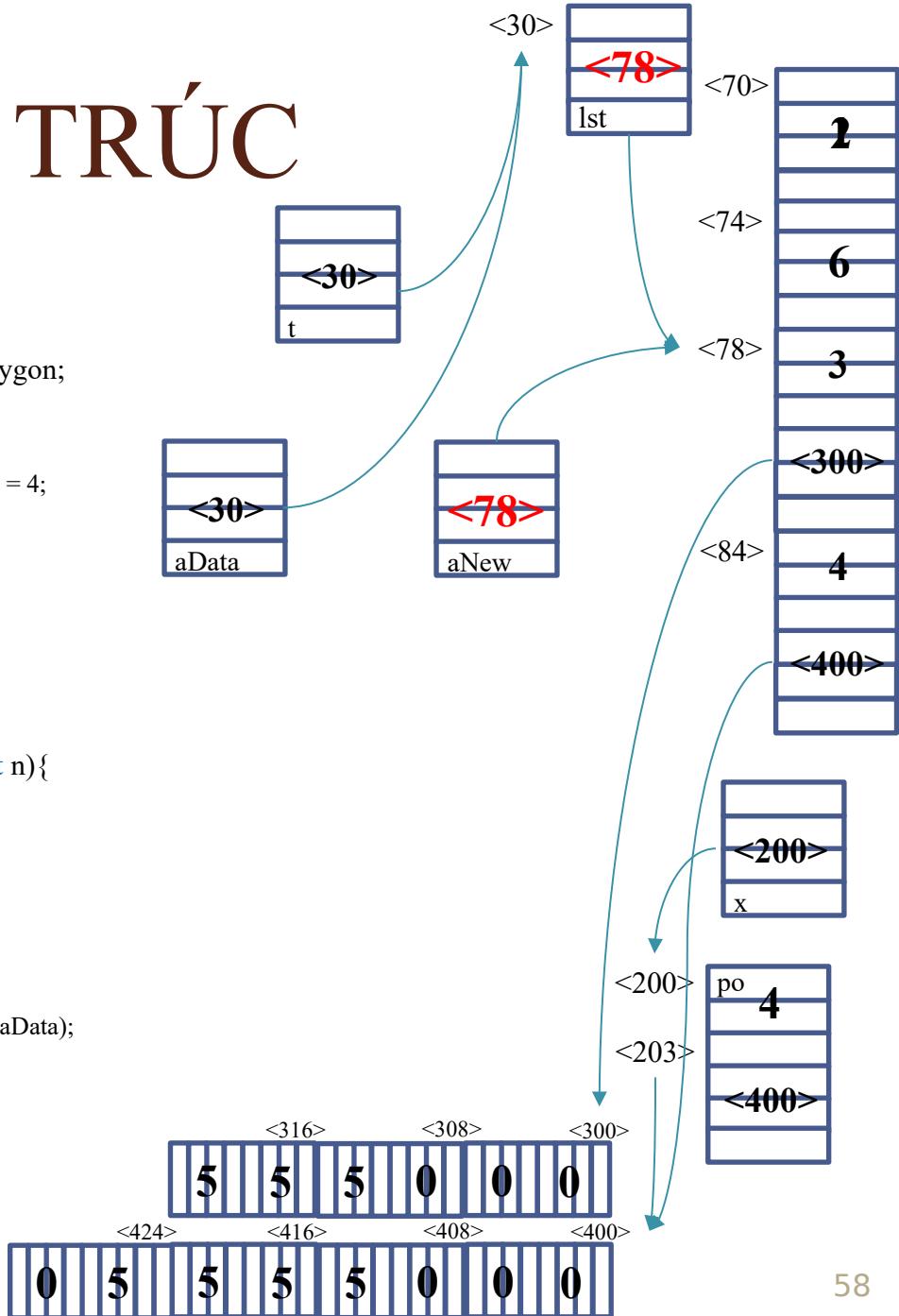
# MẢNG CẤU TRÚC

- Các khai báo struct cần thiết
  - `typedef struct{ int x, y; } Point;`
  - `typedef struct { short nVer; Point* Vers; } Polygon;`
  - `void main()`
    - `Point dg1[] = {{0, 0}, {0, 5}, {5, 5}}; int n1 = 3;`
    - `...`
    - `Polygon* lst = PolyListInit();`
    - `if(lst != NULL){`
    - `PolyListPush(&lst, dg1, n1);`
    - `...`
    - `}`
  - `int PolyListPush(Polygon** t, Point* P, short n){`
    - `Polygon po = {n};`
    - `po.Vers = (Point*)calloc(n, sizeof(Point));`
    - `for(int i = 0; i < n; i++) po.Vers[i] = P[i];`
    - `return arrPushback((void**)t, (void*)&po);`
  - `int arrPushback(void*** aData, void* x){`
    - `int nItem = arrSize(*aData), szItem = arrItemSize(*aData);`
    - `void* aNew = arrResize(*aData, 1 + nItem);`
    - `if(aNew != NULL){`
    - `memmove((char*)aNew + nItem * szItem, x, szItem);`
    - `*aData = aNew;`
    - `return 1;`
  - `}`
  - `return 0;`



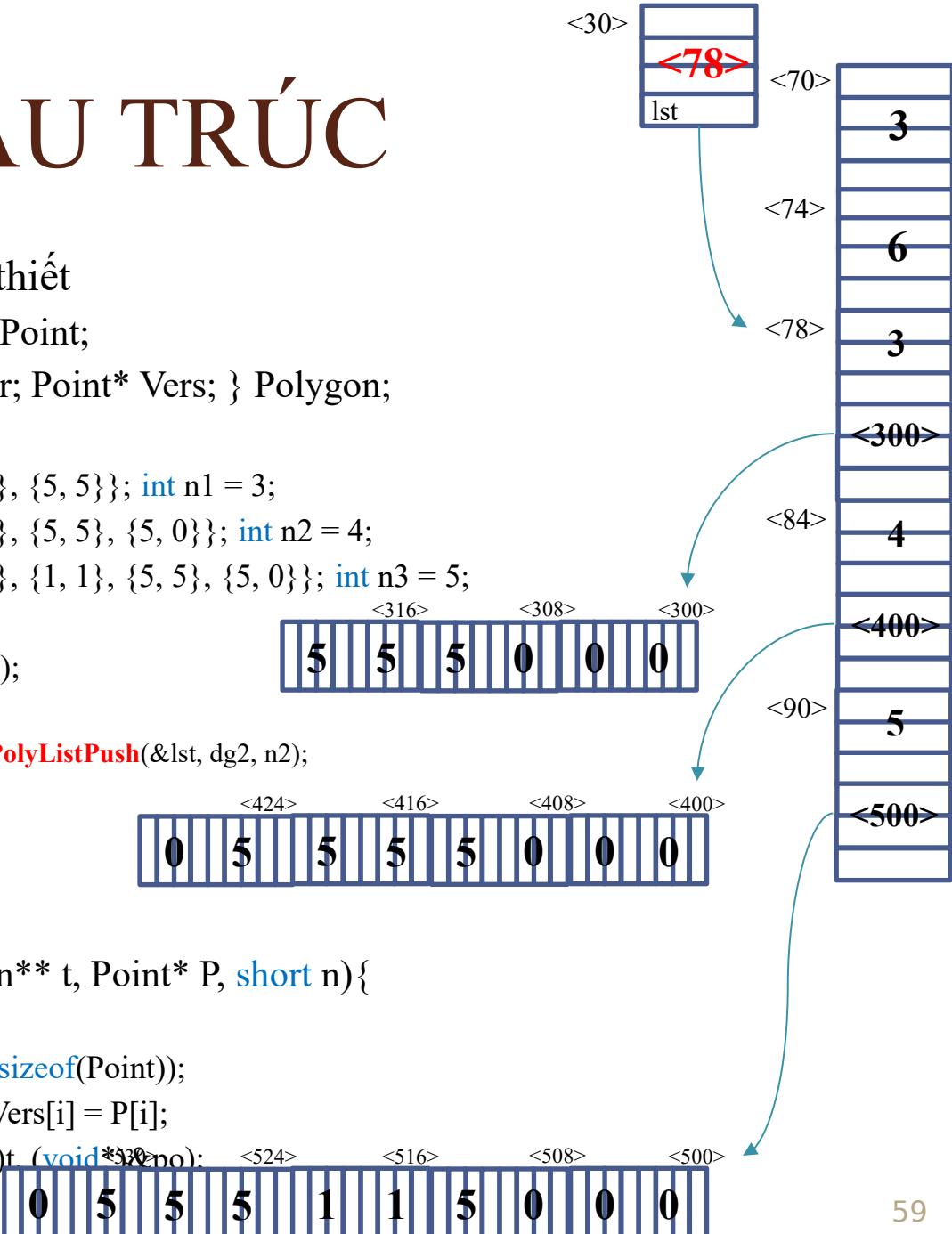
# MẢNG CẤU TRÚC

- Các khai báo struct cần thiết
  - `typedef struct{ int x, y; } Point;`
  - `typedef struct { short nVer; Point* Vers; } Polygon;`
  - `void main(){`
    - `Point dg1[] = {{0, 0}, {0, 5}, {5, 5}}; int n1 = 3;`
    - `Point dg2[] = {{0, 0}, {0, 5}, {5, 5}, {5, 0}}; int n2 = 4;`
    - `...`
    - `Polygon* lst = PolyListInit();`
    - `if(lst != NULL){`
      - `PolyListPush(&lst, dg1, n1); PolyListPush(&lst, dg2, n2);`
      - `...`
      - `}`
    - `}`
    - `int PolyListPush(Polygon** t, Point* P, short n){`
      - `Polygon po = {n};`
      - `po.Vers = (Point*)calloc(n, sizeof(Point));`
      - `for(int i = 0; i < n; i++) po.Vers[i] = P[i];`
      - `return arrPushback((void**)t, (void*)&po);`
    - `}`
    - `int arrPushback(void** aData, void* x){`
      - `int nItem = arrSize(*aData), szItem = arrItemSize(*aData);`
      - `void* aNew = arrResize(*aData, 1 + nItem);`
      - `if(aNew != NULL){`
        - `memmove((char*)aNew + nItem * szItem, x, szItem);`
        - `*aData = aNew;`
        - `return 1;`
      - `}`
      - `return 0;`



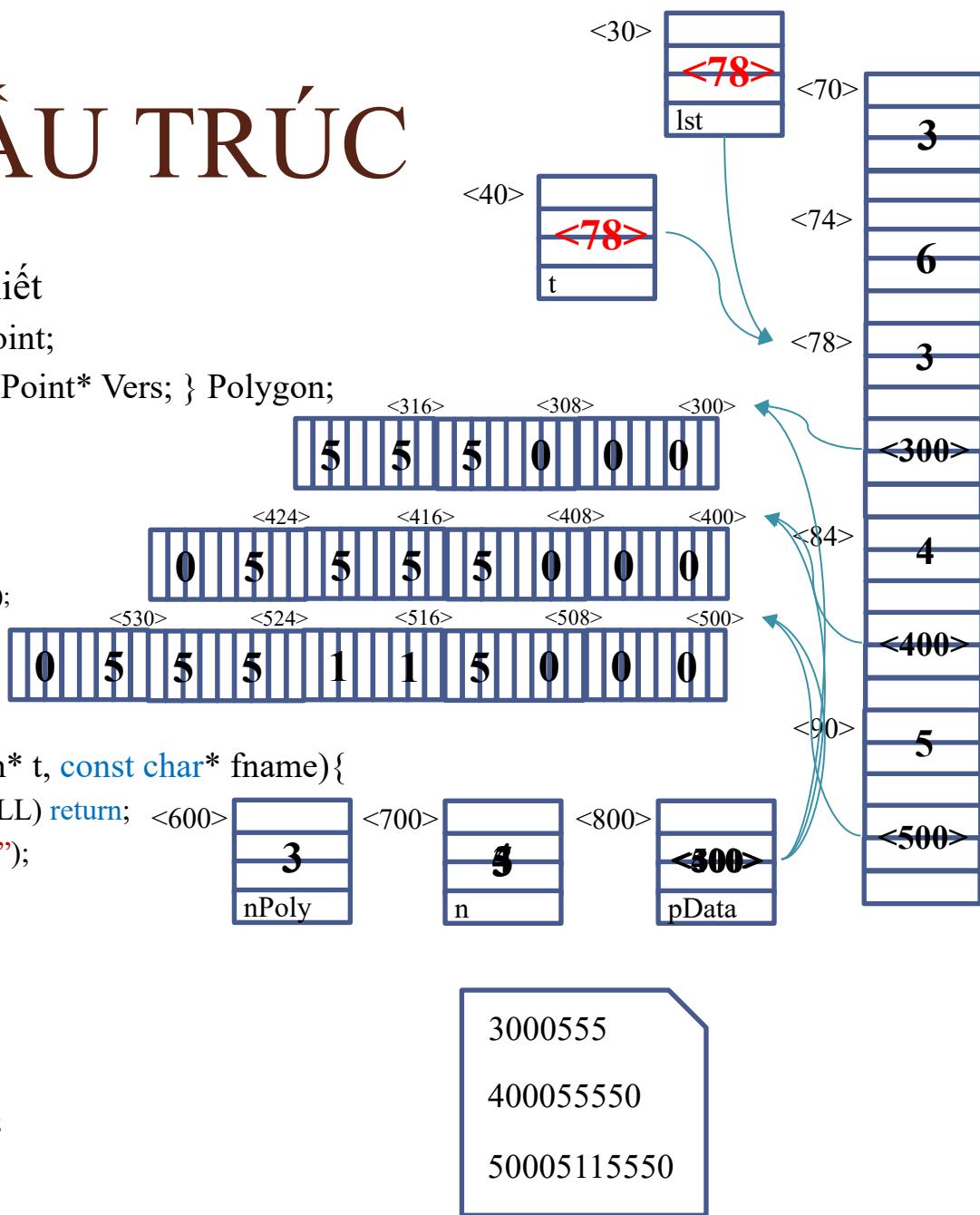
# MẢNG CẤU TRÚC

- Các khai báo struct cần thiết
  - `typedef struct{ int x, y; } Point;`
  - `typedef struct { short nVer; Point* Vers; } Polygon;`
  - `void main(){`
  - `Point dg1[] = {{0, 0}, {0, 5}, {5, 5}}; int n1 = 3;`
  - `Point dg2[] = {{0, 0}, {0, 5}, {5, 5}, {5, 0}}; int n2 = 4;`
  - `Point dg3[] = {{0, 0}, {0, 5}, {1, 1}, {5, 5}, {5, 0}}; int n3 = 5;`
  - `//...`
  - `Polygon* lst = PolyListInit();`
  - `if(lst != NULL){`
  - `PolyListPush(&lst, dg1, n1); PolyListPush(&lst, dg2, n2);`
  - `PolyListPush(&lst, dg3, n3);`
  - `//...`
  - `}`
  - `}`
  - `int PolyListPush(Polygon** t, Point* P, short n){`
  - `Polygon po = {n};`
  - `po.Vers = (Point*)calloc(n, sizeof(Point));`
  - `for(int i = 0; i < n; i++) po.Vers[i] = P[i];`
  - `return arrPushback((void**)t, (void*)&po);`
  - `}`



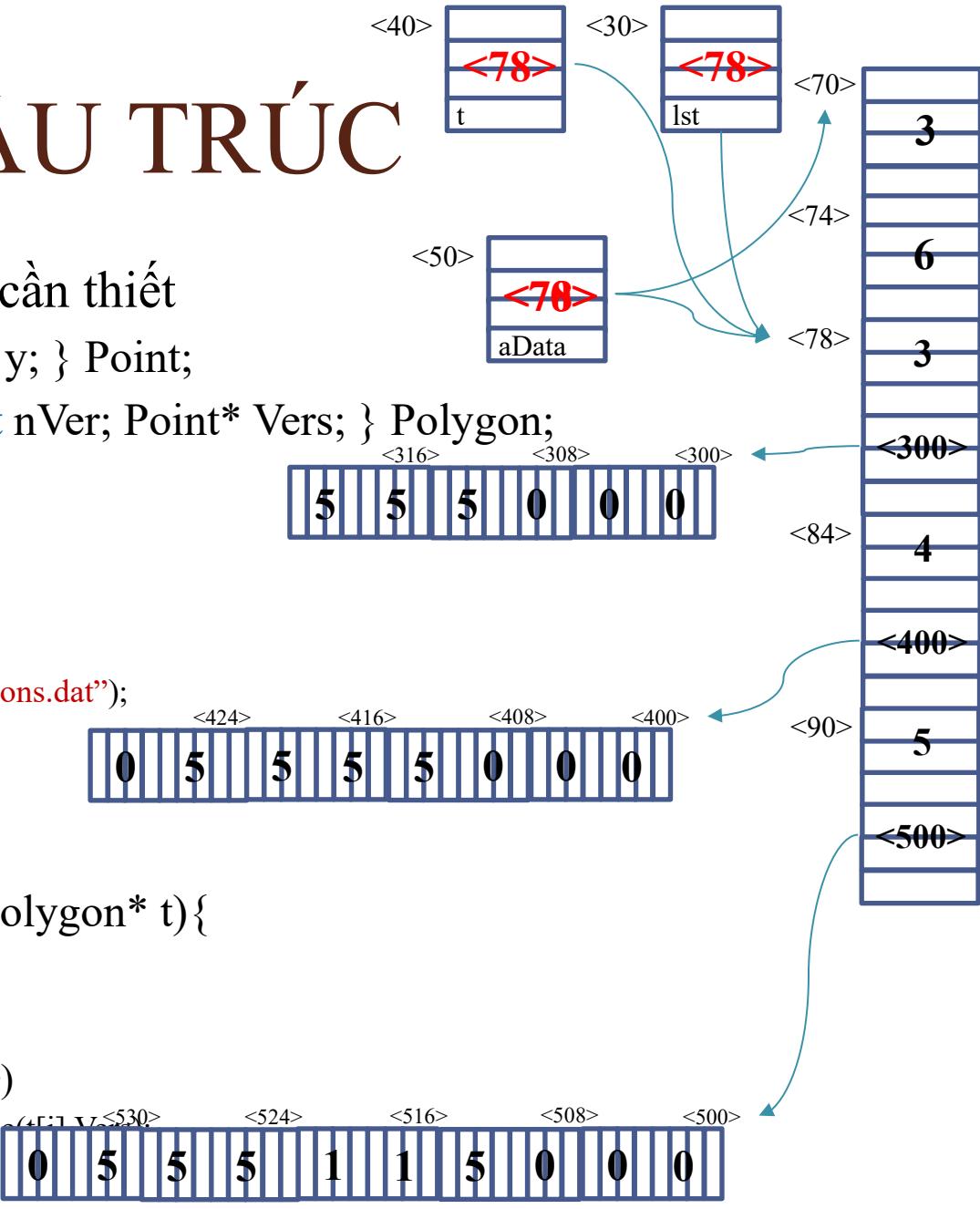
# MẢNG CẤU TRÚC

- Các khai báo struct cần thiết
  - `typedef struct{ int x, y; } Point;`
  - `typedef struct { short nVer; Point* Vers; } Polygon;`
  - `void main(){`
    - `//...`
    - `if(lst != NULL){`
      - `//...`
      - `PolyListSave(lst, "Polygons.dat");`
      - `PolyListFree(lst);`
    - `}`
  - `}`
  - `void PolyListSave(Polygon* t, const char* fname){`
    - `if(t == NULL || fname == NULL) return;`
    - `FILE* fp = fopen(fname, "wb");`
    - `if(fp == NULL) return;`
    - `int nPoly = arrSize(t);`
    - `for(int i = 0; i < nPoly; i++){`
      - `short n = t[i].nVer;`
      - `void* pData = t[i].Vers;`
      - `fwrite(&n, sizeof(n), 1, fp);`
      - `fwrite(pData, sizeof(Point), n, fp);`
    - `}`
    - `fclose(fp);`



# MẢNG CẤU TRÚC

- Các khai báo struct cần thiết
  - `typedef struct{ int x, y; } Point;`
  - `typedef struct { short nVer; Point* Vers; } Polygon;`
  - `void main(){`
    - `//...`
    - `if(lst != NULL){`
      - `//...`
      - `PolyListSave(lst, "Polygons.dat");`
      - `PolyListFree(lst);`
    - `}`
  - `}`
  - `void PolyListFree(Polygon* t){`
    - `if(t == NULL) return;`
    - `int n = arrSize(t);`
    - `for(int i = 0; i < n; i++)`
      - `if(t[i].Vers != NULL) free(t[i].Vers);`
    - `arrFree(t);`
  - `}`



# MẢNG CẤU TRÚC

- Xây dựng mảng cấu trúc các đa giác để **đọc lại file nhị phân vừa tạo**
- Có hai cách xây dựng
  - Tạo mới **struct** với kỹ thuật mảng một phần tử
  - Tái sử dụng **Polygon\***

Có m đa giác

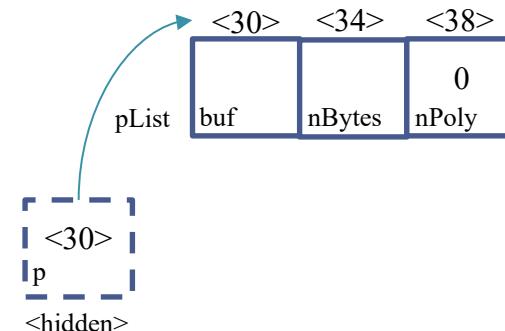


```
<số đỉnh n1><x1><y1>...<xn_1><yn_1>
<số đỉnh n2><x1><y1>...<xn_2><yn_2>
...
<số đỉnh nm><x1><y2>...<xn_m><yn_m>
```

# MẢNG CẤU TRÚC

- Các khai báo struct cần thiết (cách 1)

- ```
typedef struct { short nVer; Point Vers[1]; } PolygonDat;
```
- ```
typedef struct {
    char* buf; // Dữ liệu
    long nBytes; // Kích thước file
    int nPoly; // Số lượng đỉnh
    PolygonDat* operator[](int i){ //... }
}
```
- ```
PolygonList;
void main(){
    //phần tạo tập tin như phần trước
    PolygonList pList; InitPolyList(pList, "Polygons.dat");
    if(pList.buf != NULL){
        for(int i = 0; i < pList.nPoly; i++){
            PolygonDat* pg = pList[i];
            ShowPoly(pg); cout << endl;
        }
        free(pList.buf);
    }
}
```
- ```
void InitPolyList(PolygonList& p, const char* fname){
    p.nPoly = 0;
    p.buf = PolyRead(fname, p.nBytes);
    CountPoly(p);
}
```



# MẢNG CẤU TRÚC

- Các khai báo struct cần thiết (cách 1)

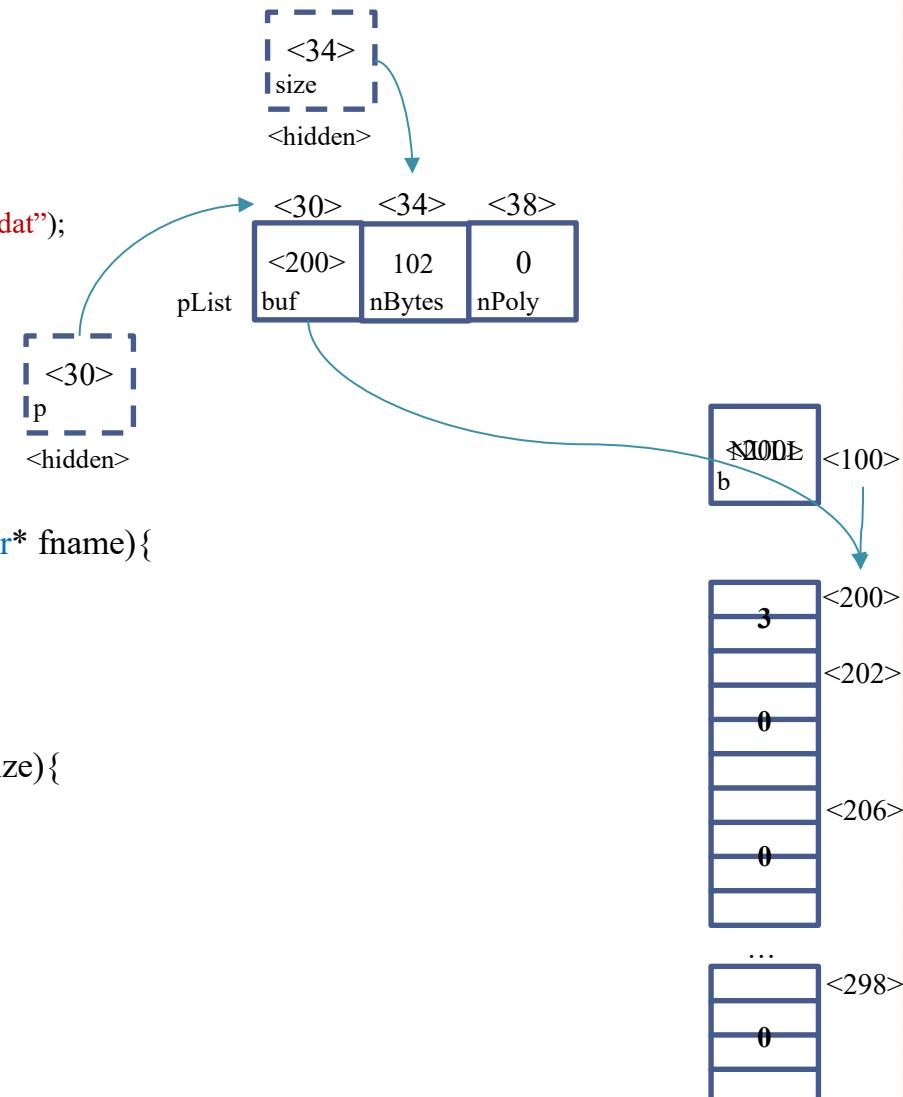
```

• void main(){
    • //phản tạo tập tin như phần trước
    • PolygonList pList; InitPolyList(pList, "Polygons.dat");
    • if(pList.buf != NULL){
        • for(int i = 0; i < pList.nPoly; i++){
            • PolygonDat* pg = pList[i];
            • ShowPoly(pg); cout << endl;
            • free(pList.buf);
        }
    }
    • }

• void InitPolyList(PolygonList& p, const char* fname){
    • p.nPoly = 0;
    • p.buf = PolyRead(fname, p.nBytes);
    • CountPoly(p);
}

• char* PolyRead(const char* fname, long& size){
    • char* b = NULL;
    • FILE* fp = fopen(fname, "rb");
    • if(fp != NULL){
        • fseek(fp, 0, SEEK_END); size = ftell(fp);
        • fseek(fp, 0, SEEK_SET);
        • b = (char*)malloc(size);
        • if(b != NULL) fread(b, size, 1, fp);
        • fclose(fp);
    }
    • return b;
}

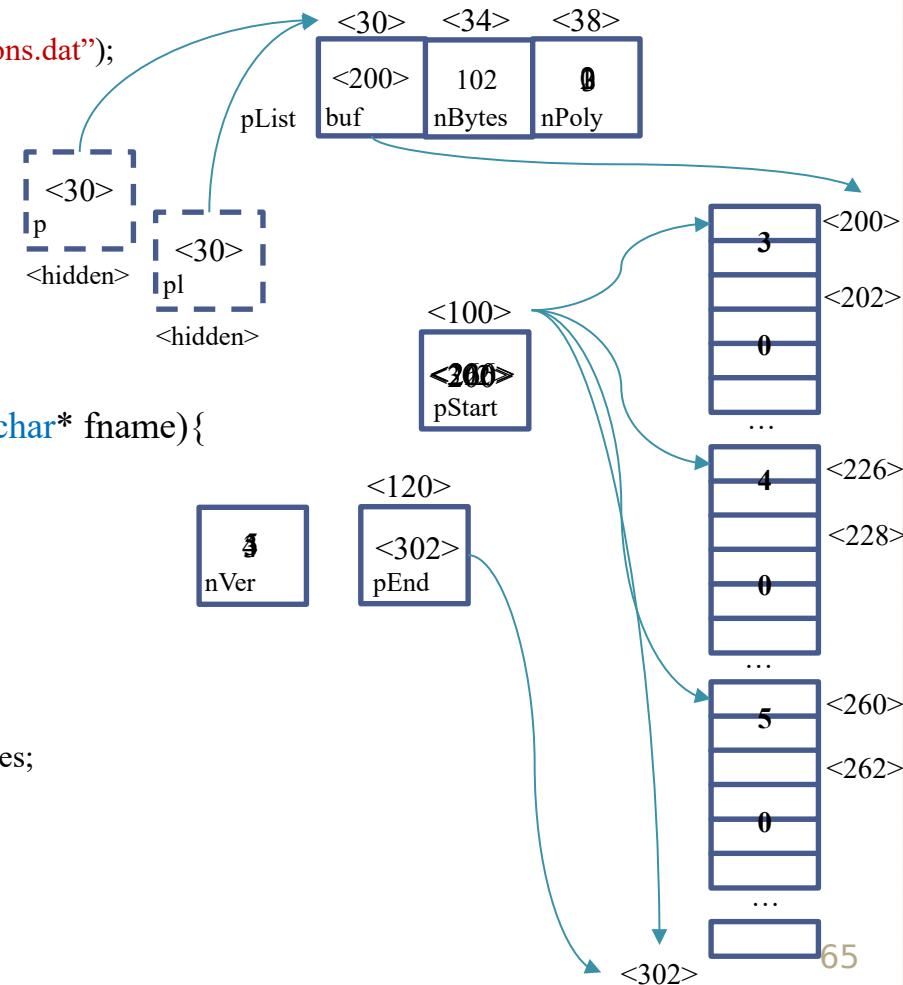
```



# MẢNG CẤU TRÚC

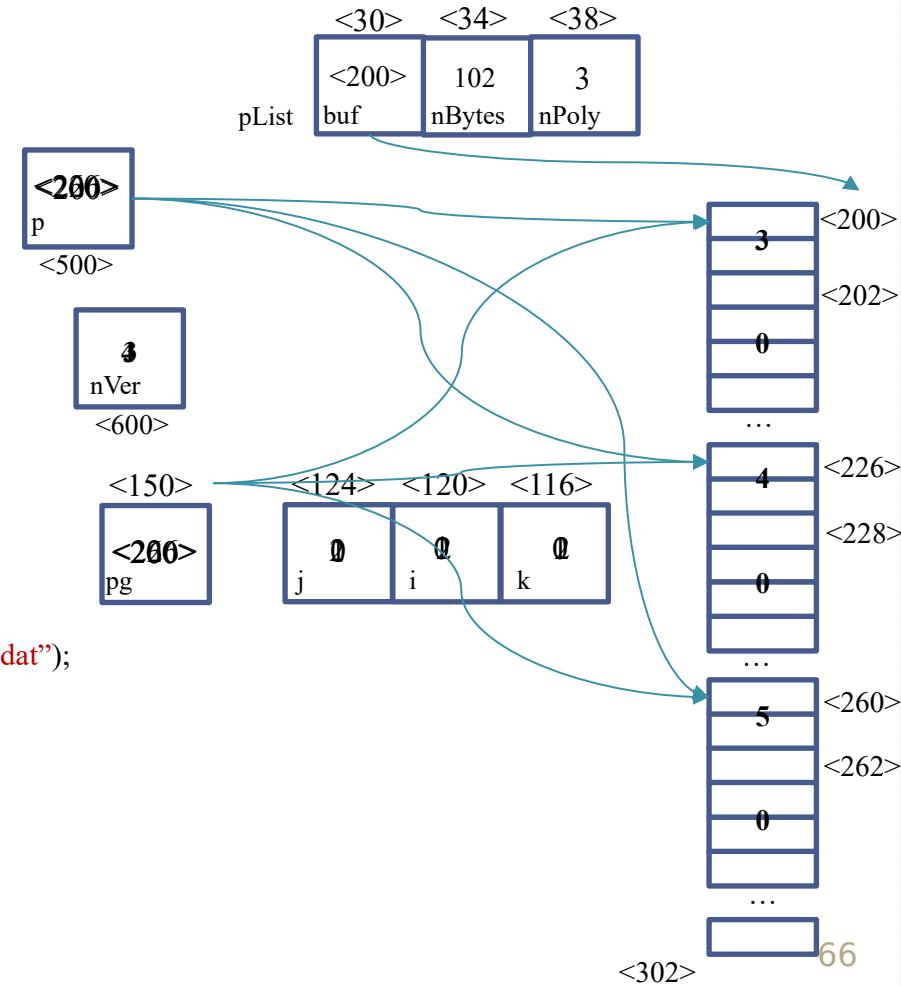
- Các khai báo struct cần thiết (cách 1)

```
• void main(){  
    //phần tạo tập tin như phần trước  
    PolygonList pList; InitPolyList(pList, "Polygons.dat");  
    if(pList.buf != NULL){  
        for(int i = 0; i < pList.nPoly; i++){  
            PolygonDat* pg = pList[i];  
            ShowPoly(pg); cout << endl;  
        }  
        free(pList.buf);  
    }  
}  
  
• void InitPolyList(PolygonList& p, const char* fname){  
    p.nPoly = 0;  
    p.buf = PolyRead(fname, p.nBytes);  
    CountPoly(p);  
}  
  
• void CountPoly(PolygonList& pl){  
    if(pl.buf == NULL) return;  
    char* pStart = pl.buf, *pEnd = pl.buf + pl.nBytes;  
    while(pStart < pEnd){  
        short nVer = *(short*)pStart;  
        if(nVer > 0) pl.nPoly++;  
        pStart += sizeof(short) + nVer * sizeof(Point);  
    }  
}
```



# MẢNG CẤU TRÚC

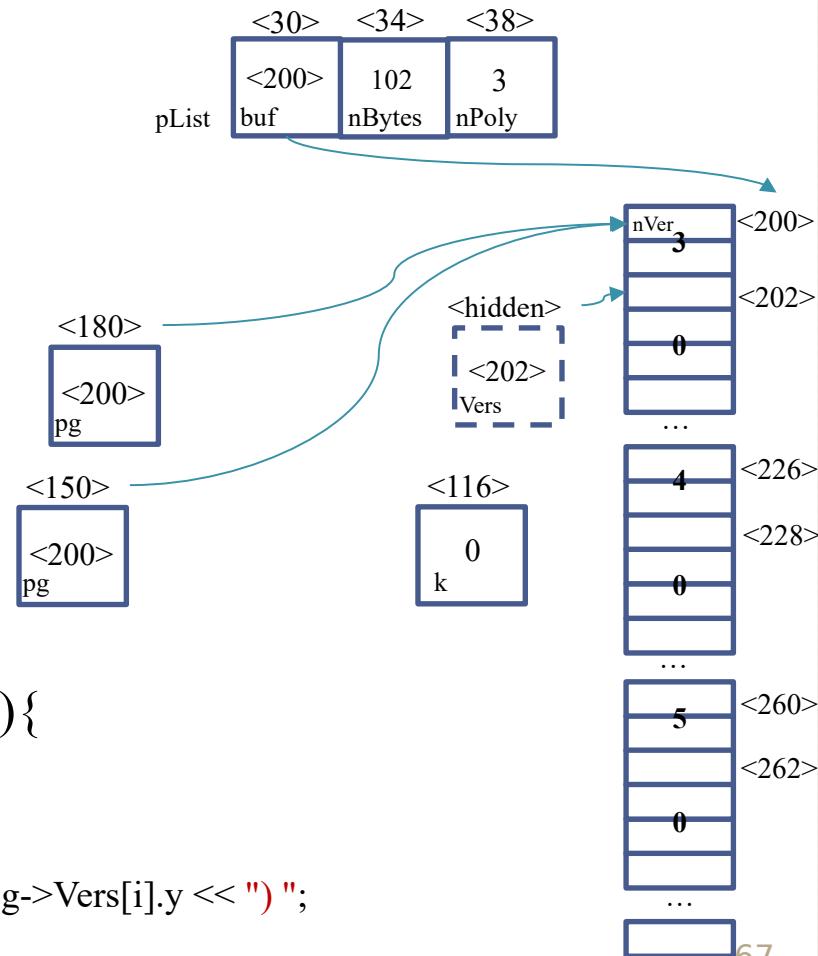
- Các khai báo struct cần thiết (cách 1)
  - `typedef struct { short nVer; Point Vers[1]; } PolygonDat;`
  - `typedef struct {
 char* buf; // Dữ liệu
 long nBytes; // Kích thước file
 int nPoly; // Số lượng đỉnh
 }`
  - `PolygonDat* operator[](int i){
 int j = 0; char* p = buf;
 while(j < i){
 short nVer = *(short*)p;
 p += sizeof(short) + nVer * sizeof(Point);
 j++;
 }
 return (PolygonDat*)p;
 }`
  - `} PolygonList;`
  - `void main(){
 //phản tạo tập tin như phần trước
 PolygonList pList; InitPolyList(pList, "Polygons.dat");
 if(pList.buf != NULL){
 for(int k = 0; k < pList.nPoly; k++){
 PolygonDat* pg = pList[k];
 ShowPoly(pg); cout << endl;
 }
 free(pList.buf);
 }
 }`



# MẢNG CẤU TRÚC

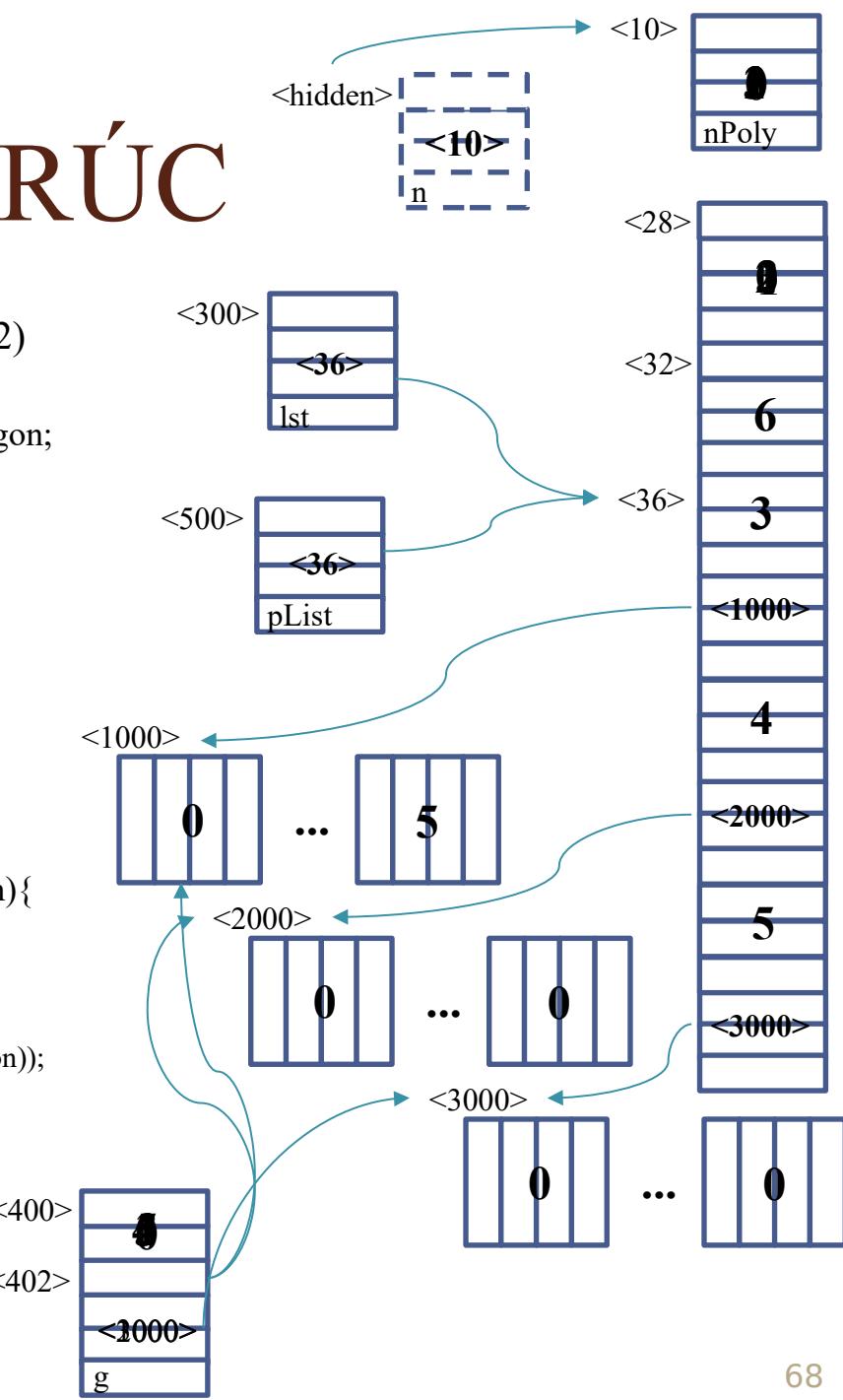
- Các khai báo struct cần thiết (cách 1)

- `void main(){`
  - //phản tạo tập tin như phần trước
  - `PolygonList pList;`
  - `InitPolyList(pList, “Polygons.dat”);`
  - `if(pList.buf != NULL){`
    - `for(int i = 0; i < pList.nPoly; i++){`
      - `PolygonDat* pg = pList[i];`
      - **ShowPoly(pg); cout << endl;**
    - `}`
    - `free(pList.buf);`
  - `}`
- `}`
- `void ShowPoly(PolygonDat* pg){`
  - `cout << pg->nVer << " đỉnh: ";`
  - `for(int i = 0; i < pg->nVer; i++)`
    - `cout << "(" << pg->Vers[i].x << ", " << pg->Vers[i].y << ") ";`
- `}`



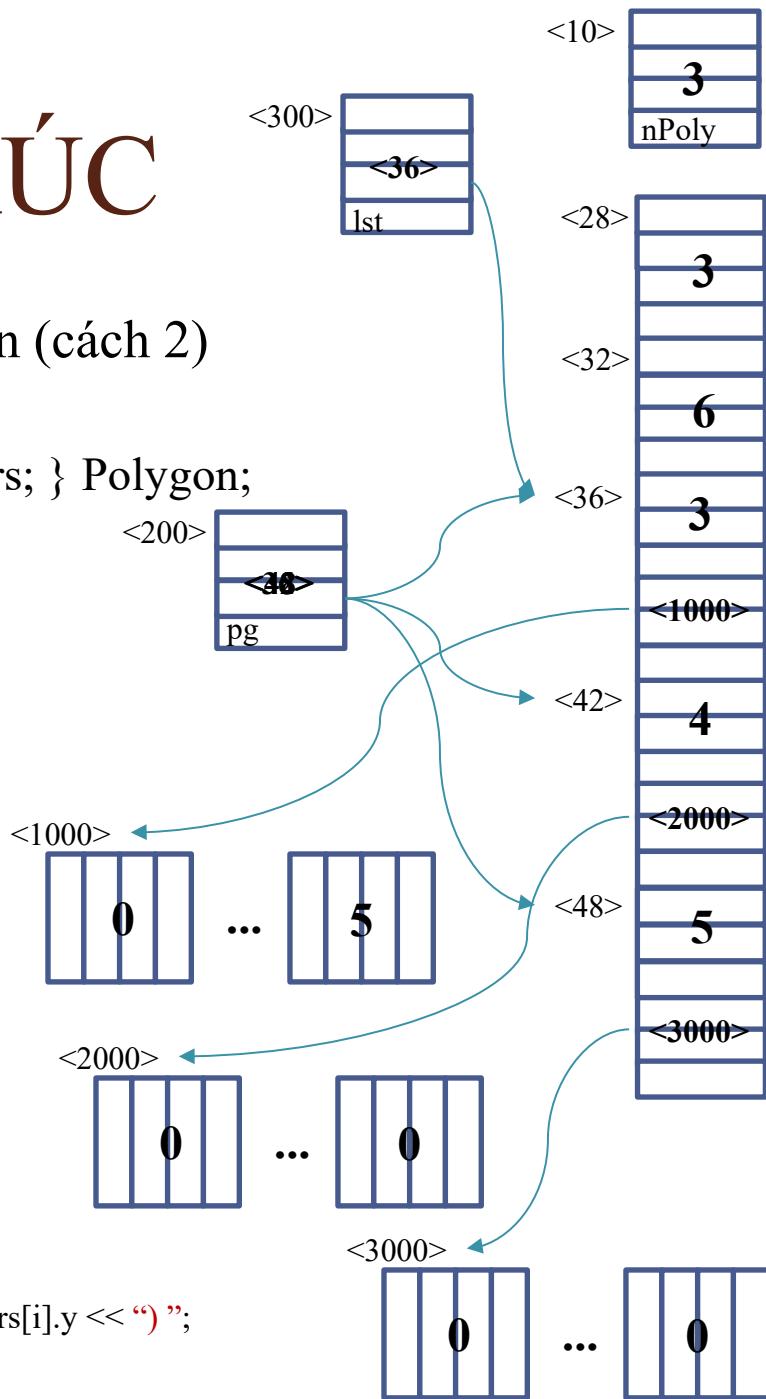
# MÀNG CẤU TRÚC

- Tái sử dụng **struct Point & Polygon** (cách 2)
    - **typedef struct{ int x, y; } Point;**
    - **typedef struct {short nVer; Point\* Vers; } Polygon;**
    - **void main()**{
      - //phần tạo tập tin như phần trước
      - **int nPoly;**
      - **lst = PolyRead("Polygons.dat", nPoly);**
      - **if(lst){**
        - **for(int i = 0; i < nPoly; i++){**
          - **ShowPoly(&lst[i]); cout << endl;**
          - **}**
        - **} PolyListFree(lst);**
      - }
    - **Polygon\* PolyRead(const char\* fname, int& n){**
      - **FILE\* fp = fopen(fname, "rb");**
      - **if(fp == NULL) return NULL;**
      - **n = 0;**
      - **Polygon\* pList = (Polygon\*)arrInit(0, sizeof(Polygon));**
      - **while(!feof(fp)){**
        - **Polygon g = {0};**
        - **if(fread(&(g.nVer), sizeof(g.nVer), 1, fp) <= 0) break;**
        - **g.Vers = (Point\*)malloc(g.nVer \* sizeof(Point));** <400>
        - **if(g.Vers != NULL){**
          - **n++;** <402>
          - **fread(g.Vers, g.nVer, sizeof(Point), fp);**
          - **arrPushback((void\*\*)&pList, (void\*)&g);**
        - **}**
      - **} fclose(fp);**



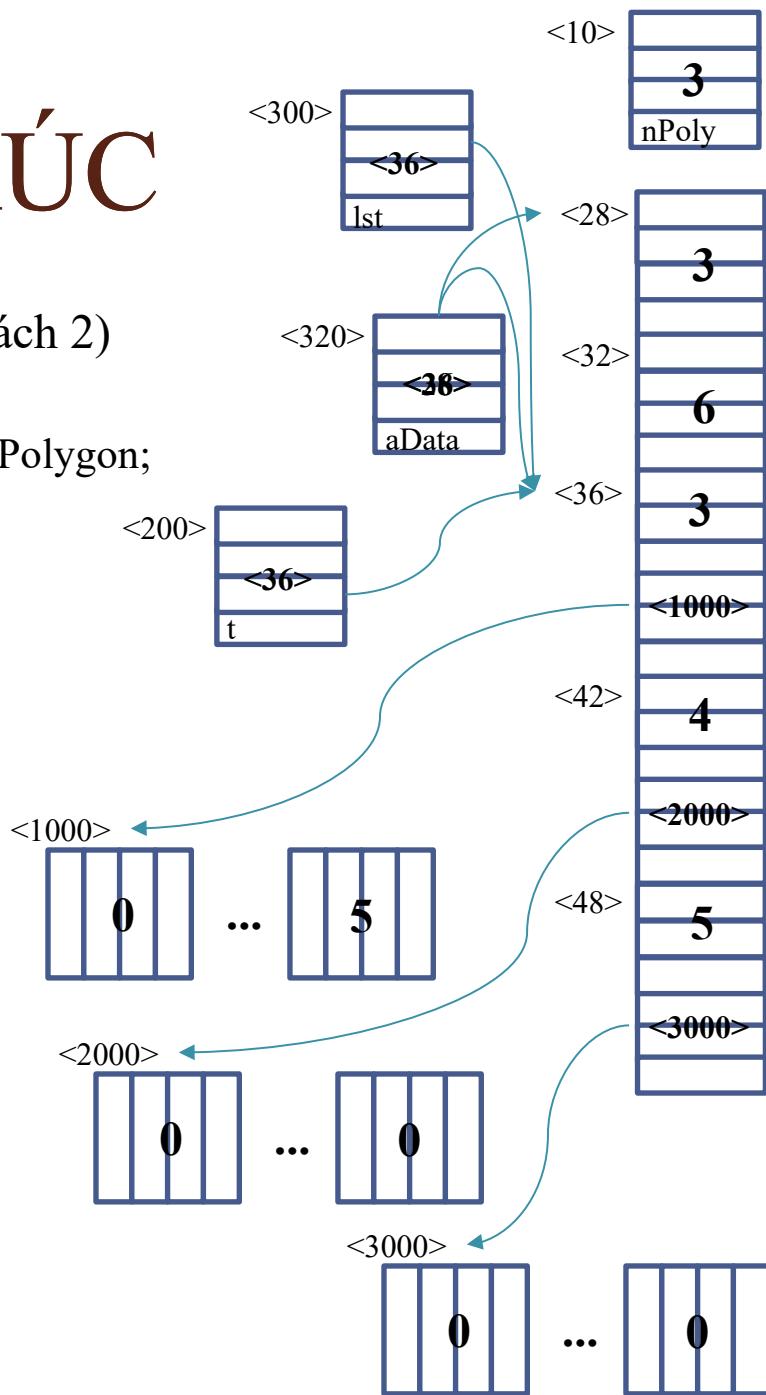
# MẢNG CẤU TRÚC

- Tái sử dụng **struct Point & Polygon** (cách 2)
  - typedef struct{ int x, y; } Point;**
  - typedef struct {short nVer; Point\* Vers; } Polygon;**
  - void main()**{
  - //**phân tạo tập tin như phần trước**
  - int nPoly;**
  - lst = PolyRead("Polygons.dat", nPoly);**
  - if(lst){**
  - **for(int i = 0; i < nPoly; i++){**
  - **ShowPoly(&lst[i]); cout << endl;**
  - **}**
  - **}**
  - PolyListFree(lst);**
  - }**
  - void ShowPoly(Polygon\* pg){**
  - **cout << pg->nVer << " dinh: ";**
  - **for(int i = 0; i < pg->nVer; i++)**
  - **cout << "(" << pg->Vers[i].x << "," << pg->Vers[i].y << ") ";**
  - }**



# MẢNG CẤU TRÚC

- Tái sử dụng **struct Point & Polygon** (cách 2)
  - `typedef struct{ int x, y; } Point;`
  - `typedef struct {short nVer; Point* Vers; } Polygon;`
  - `void main(){`
    - //phản tạo tập tin như phần trước**
    - `int nPoly;`
    - `lst = PolyRead("Polygons.dat", nPoly);`
    - `if(lst){`
      - `for(int i = 0; i < nPoly; i++){`
        - `ShowPoly(&lst[i]); cout << endl;`
    - `}`
    - `}`
  - PolyListFree(lst);**
  - `}`
  - `void PolyListFree(Polygon* t){`
    - `if(t == NULL) return;`
    - `int n = arrSize(t);`
    - `for(int i = 0; i < n; i++)`
      - `if(t[i].Vers != NULL) free(t[i].Vers);`
    - `arrFree(t);`
  - `}`



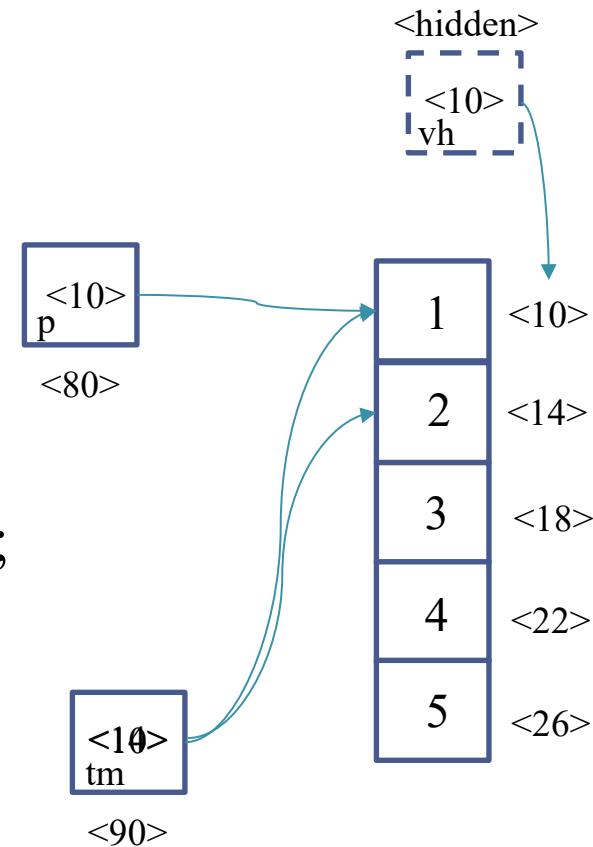
# KỸ THUẬT NÂNG CAO (CON TRỎ VÔ KIẾU)

- Là con trỏ có kiểu là **void**, ví dụ **void\*** p
- Có thể dùng trong những trường hợp ta chưa xác định được kiểu dữ liệu thực tế của dữ liệu
- Hạn chế chính là không thể dùng các toán tử thông thường trên con trỏ để tính toán. Ví dụ không thể viết  $p + 1$  hay  $*(p - 1)$  vì  $p$  kiểu **void\*** nên không thể tính bước nhảy
- Dùng kỹ thuật template trong C++ để thay thế

# KỸ THUẬT NÂNG CAO (CON TRỎ VÔ KIẾU)

- Ví dụ:

- `void main(){`
  - `int a[] = {1, 2, 3, 4, 5};`
  - `void* p = a;`
  - `cout << p[1] << endl; // Sai`
  - `cout << *((int*)p + 1) << endl;`
  - `cout << ((int*)p)[1] << endl;`
- `}`

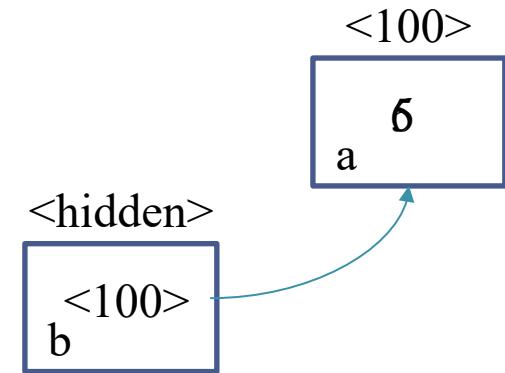


# KỸ THUẬT NÂNG CAO (BIẾN THAM CHIẾU)

- Là biến tham chiếu tới một biến khác
- Biến tham chiếu cần có biến gốc, không thể tồn tại độc lập
- Có ba cách dùng biến tham chiếu chính
  - Khai báo biến tham chiếu (cùng kiểu) tới biến gốc trong cùng phạm vi, ví dụ: `int a = 5; int& b = a;`
  - Là tham số tham chiếu trong hàm, ví dụ `Func(<kiểu>& x)`
  - Giá trị trả về của hàm là một biến tham chiếu, ví dụ `<kiểu>& Func()`. Lưu ý biến gốc phải còn tồn tại khi biến tham chiếu trả về (không tham chiếu tới biến cục bộ trong hàm)
- Có thể dùng tham chiếu cho con trỏ

# KỸ THUẬT NÂNG CAO (BIẾN THAM CHIẾU)

- Khai báo biến tham chiếu (cùng kiểu) tới biến gốc trong cùng phạm vi
- Ví dụ
  - `void main(){`
    - `int a = 5;`
    - `int& b = a;`
    - `a = 6;`
    - `cout << "a la: " << a << endl;`
    - `cout << "b la: " << b << endl;`
    - `cout << "Dia chi cua a la: " << &a << endl;`
    - `cout << "Dia chi cua b la: " << &b << endl;`
  - `}`



# KỸ THUẬT NÂNG CAO (DỮ LIỆU & CON TRỎ HẰNG)

- Con trỏ có thể “hằng hóa” vùng nhớ dữ liệu
  - `void main(){`
    - `char a[] = "Hello world!!!"; const char* pstr;`
    - `a[0] = 'h'; // Đúng`
    - `pstr = a;`
    - `pstr[0] = 'H'; // Sai      không thay đổi nd`
    - `a[0] = 'H' // Đúng`
    - `pstr = "a another string"; // Đúng`
  - `}`
- `pstr` có thể trỏ tới địa chỉ khác
- Lưu ý:
  - Câu lệnh `pstr = (char*)pstr` chạy được nhưng vô nghĩa vì dữ liệu cũng không thể thay đổi với con trỏ này. Ví dụ câu lệnh `pstr[0] = 'H'` vẫn sẽ báo lỗi.
  - Có thể ép kiểu nhưng phải trả ra con trỏ khác. Ví dụ `char* tmp = (char*)pstr;` Lúc này có thể dùng `tmp` để hiệu chỉnh chuỗi

# KỸ THUẬT NÂNG CAO (DỮ LIỆU & CON TRỎ HẰNG)

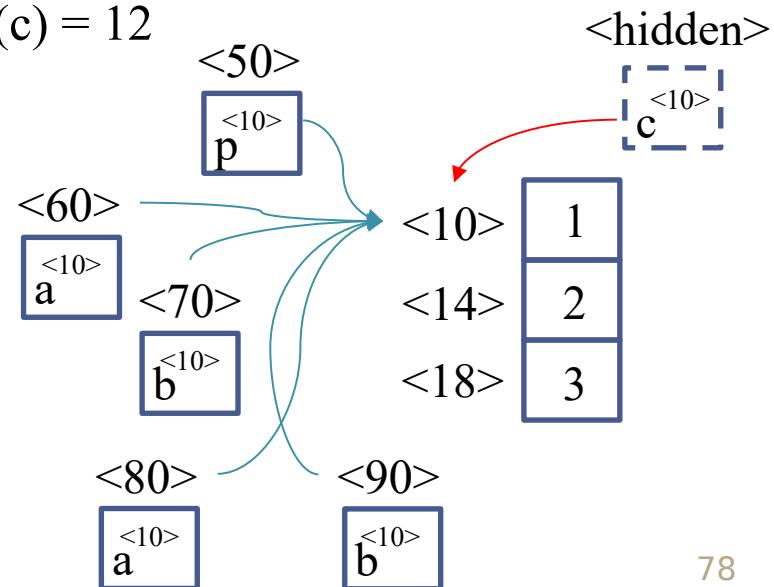
- **Con trả hằng** là con trả chỉ chứa duy nhất một địa chỉ lúc khai báo và định nghĩa
  - `void main(){`
  - `char a[] = "Hello world!!!";`
  - `char* const pstr = a; //Khai báo & đ/nghĩa cùng lúc`
  - `pstr[0] = 'h'; // Đúng`
  - `*(pstr + 1) = 'E'; // Đúng`
  - `pstr = pstr + 1; // Sai`
  - `}`
- Có thể **thay đổi giá trị** cho **con trả hằng** bằng kỹ thuật ép kiểu kết hợp tham chiếu
  - `void main(){`
  - `char a[] = "Hello world!!!"; char* const pstr = a;`
  - `pstr[0] = 'h'; // Đúng`
  - `*(pstr + 1) = 'E'; // Đúng`
  - `char* &tmp = (char*&)pstr; tmp++;`
  - `cout << pstr[0] << endl; // in ra chữ 'E'`
  - `}`
- Lưu ý: dùng trực tiếp tên biến pstr **vẫn bị lỗi**. Ví dụ câu lệnh `pstr++` và `pstr = (char*)pstr` là sai

# KỸ THUẬT NÂNG CAO (DỮ LIỆU & CON TRỎ HẰNG)

- Ngoài **Con trả hằng**, có thể dùng **const** cho các kiểu phi con trả dạng cấu trúc và cơ sở để “hằng hóa” dữ liệu
- Ví dụ:
  - Point **const** P = {0, 1}; **const** Point P = {0, 1};
  - **int const** a[] = {1, 2}; **const int** a[] = {1, 2};
- Lưu ý: với **hai** cách viết, **a** và **&P** lần lượt có kiểu là **const int\*** và **const Point\***
- Việc thay đổi giá trị P.x, P.y và các phần tử mảng a là bất hợp lệ
- Có thể áp dụng cơ chế ép kiểu trả về biến khác để **thay đổi giá trị** của các biến

# KỸ THUẬT NÂNG CAO (MẢNG TĨNH)

- Xét mảng tĩnh một chiều
  - `int c[] = {1, 2, 3}; *p = c;`
- Lưu ý:
  - `p` và `c` đều giữ một địa chỉ.
  - Câu lệnh “`c = new int`” là sai vì `c` là **con trỏ hằng**
  - `&p ≠ p = c = &c`. Có thể nói `c` là một con trỏ đặc biệt có địa chỉ bị “che”
  - `sizeof(p) = sizeof(int*) = 4 ≠ sizeof(c) = 12`
- Việc truyền tham số cho hàm
  - `void main(){`
    - `int c[] = {1, 2, 3}, *p = c;`
    - `funcA(c); funcB(c);`
    - `funcA(p); funcB(p);`
  - `}`
  - `void funcA(int a[]){ //...}`
  - `void funcB(int* b){ //...}`



# KỸ THUẬT NÂNG CAO

## (Left-value và Right-value)

- Left-value là các biểu thức bên trái phép gán “`=`”. Ví dụ: biến đơn, biến con trỏ, biến cấu trúc (truy xuất bằng ‘`.`’ và ‘`->`’), biến phần tử mảng (truy xuất bằng ‘`[]`’), biến con trỏ kết hợp toán tử ‘`*`’
- Giá trị trả về của hàm cũng có thể đóng vai biểu thức trái
  - Giá trị trả về có kiểu tham chiếu
  - Giá trị trả về có kiểu con trỏ thường / cấu trúc
- Right-value là các biểu thức bên phải của phép gán “`=`”. Mọi biểu thức trái đều có thể đóng vai biểu thức phải, ngược lại thì chưa chắc. Ví dụ biểu thức “`&x`” chỉ là rValue

# KỸ THUẬT NÂNG CAO (Left-value và Right-value)

- Giá trị trả về kiểu tham chiếu / con trỏ cơ sở
  - `int& maxRef(int& tx, int& ty){`
    - `if(tx > ty) return tx;`
    - `return ty;`
  - `}`
  - `int* minPtr(int* px, int* py){`
    - `if(*px > *py) return py;`
    - `return px;`
  - `}`
  - `int x = 99, y = 88;`
  - `void main(){`
    - `maxRef(x, y) = 9988;` 
    - `cout << x; // x = 9988`
    - `*(minPtr(&x, &y)) = 7766;`
    - `cout << y; // y = 7766`
  - `}`

# KỸ THUẬT NÂNG CAO (Left-value và Right-value)

- Giá trị trả về có kiểu con trỏ cấu trúc
  - `typedef struct { int x, y; } Point;`
  - `Point P = {16, 10}, Q = {-8, 25};`
  - `Point* PointMinX(Point* A, Point* B){`
    - `if(A->x > B->x) return B;`
    - `return A;`
  - `}`
  - `void main(){`
    - `PointMinX(&P, &Q)->x /= 2; //Q.x = -4`
    - `*PointMinX(&P, &Q) = P; //Giá trị Q và P như nhau`
  - `}`