XỬ LÝ TẬP TIN

NHẬP MÔN LẬP TRÌNH

GVHD: Trương Toàn Thịnh

NỘI DUNG

- Giới thiệu
- Quy trình lập trình tập tin
- Các kĩ thuật hỗ trợ
- Ứng dụng

GIỚI THIỆU

- Mục tiêu lập trình tập tin là để lưu trữ dữ liệu vào bộ nhớ phụ
- Có hai loại tập tin: văn bản và nhị phân
- Tập tin văn bản chứa các ký tự văn bản (mã ASCII $\geq 0\times20$)
- Tập tin văn bản có kí tự ngăn cách dòng có mã ASCII là $0\times0D$ (‘\r’) và $0\times0A$ (‘\n’)
- Tập tin **văn bản trong windows** có kí tự kết thúc (EOF) với mã $0\times1A$ (còn có tên là SUB)
- Tập tin văn bản thô mở rộng gồm các kí tự nhiều byte

GIỚI THIỆU

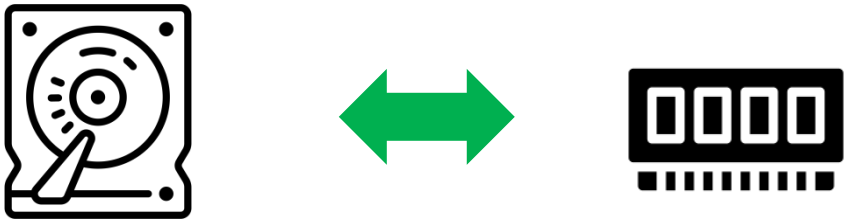
- Tập tin văn bản có thể có cấu trúc hoặc không có cấu trúc
- Tập tin nhị phân chứa dãy byte có cấu trúc (do tổ chức tạo ra quy định)
- Tập tin được lưu trong bộ nhớ phụ (ổ cứng, USB, thẻ nhớ…)
- Để truy xuất tập tin cần biết đường dẫn lưu trong bộ nhớ phụ.
- Ví dụ: “C:\\data\\list.txt” (trong C cần viết ‘\\’ thay vì ‘\’).

NỘI DUNG

- Giới thiệu
- Quy trình lập trình tập tin
- Các kĩ thuật hỗ trợ
- Ứng dụng

QUI TRÌNH LẬP TRÌNH TẬP TIN

- Các bước xử lý tập tin
  - Mở tập tin (cần có đường dẫn chính xác)
  - Sử dụng tập tin
    - Đọc dữ liệu từ bộ nhớ phụ vào RAM
    - Ghi dữ liệu từ RAM vào bộ nhớ phụ
  - Đóng tập tin (sau khi hoàn tất công việc)



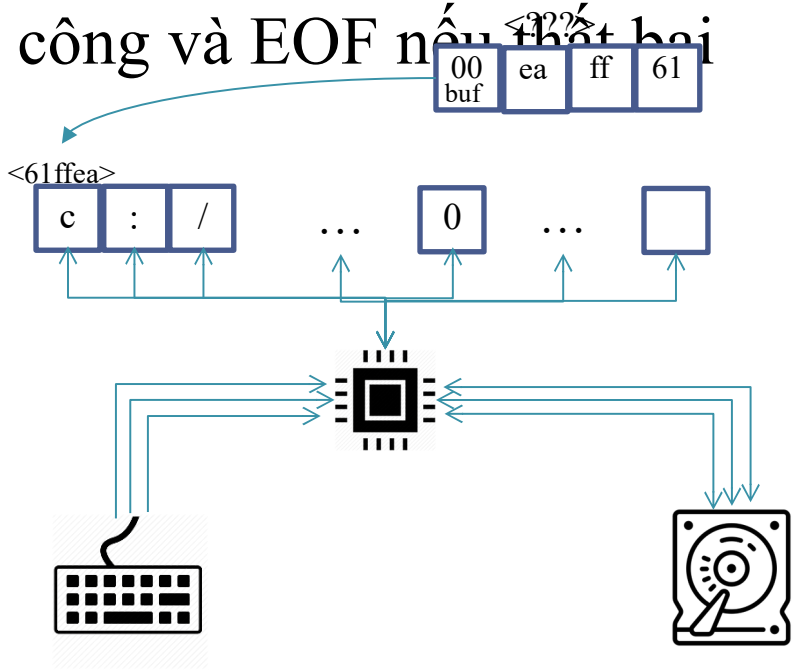
QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm mở tập tin
  - FILE* fopen(char* filename, char* mode)
- Ý nghĩa:
  - Mở tập tin có tên filename ở chế độ mode
  - Lưu ý: mode là các giá trị
    - "r", "rt": mở để đọc
    - "r+", "r+t": mở để đọc và ghi
    - "w", "wt": mở để ghi, tự tạo mới nếu chưa có tập tin, nếu đã có thì nội dung tập tin sẽ bị xóa trắng
    - "w+", "w+t": giống "w" và "wt" và thêm chức năng đọc
    - "a", "at": mở để thêm ở cuối tập tin, tạo mới nếu chưa có tập tin
    - "a+", "a+t": giống "a" và "at" và thêm chức năng đọc
- Giá trị trả về:
  - NULL nếu mở thất bại
  - FILE* nếu mở thành công

QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm đóng tập tin: FILE* fclose(FILE* fp)
- Ý nghĩa:
  - Đóng tập tin (tập tin mở thành công trước đó)
  - Dữ liệu sẽ cập nhật vào bộ nhớ phụ
- Giá trị trả về: 0 nếu thành công và EOF nếu thất bại

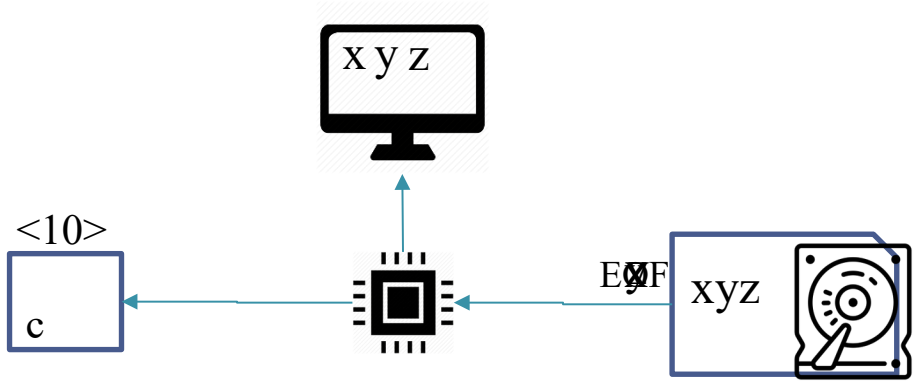
```
1  char buf[50];
2  printf("Nhap duong dan tuyet doi toi tap tin: ");
3  gets(buf); // lấy dữ liệu từ stdin, tự thêm '\0'
4  FILE* fp = fopen(buf, "r");
5  if(!fp) printf("Khong ton tai tap tin: %s\n", buf);
6  printf("Ton tai tap tin: ");
7  for(int i = 0; i < 50; i++){
8    if(*(buf + i) != '\0') printf("%c", *(buf + i));
9    else break;}
10 fclose(fp);
```



QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm đọc ký tự từ tập tin: int fgetc(FILE* fp)
- Ý nghĩa:
  - Đọc MỘT ký tự từ tập tin (tập tin mở thành công trước đó)
  - Dữ liệu đọc được sẽ lưu vào biến trong RAM
- Giá trị trả về: ký tự đọc được hoặc EOF

	Mô tả
1	FILE* fp = fopen("data.txt", "r");
2	if(fp != NULL){
3	int c = fgetc(fp);
4	while(c != EOF) {
5	printf("%c", c);
6	c = fgetc(fp);
7	}
8	fclose(fp);

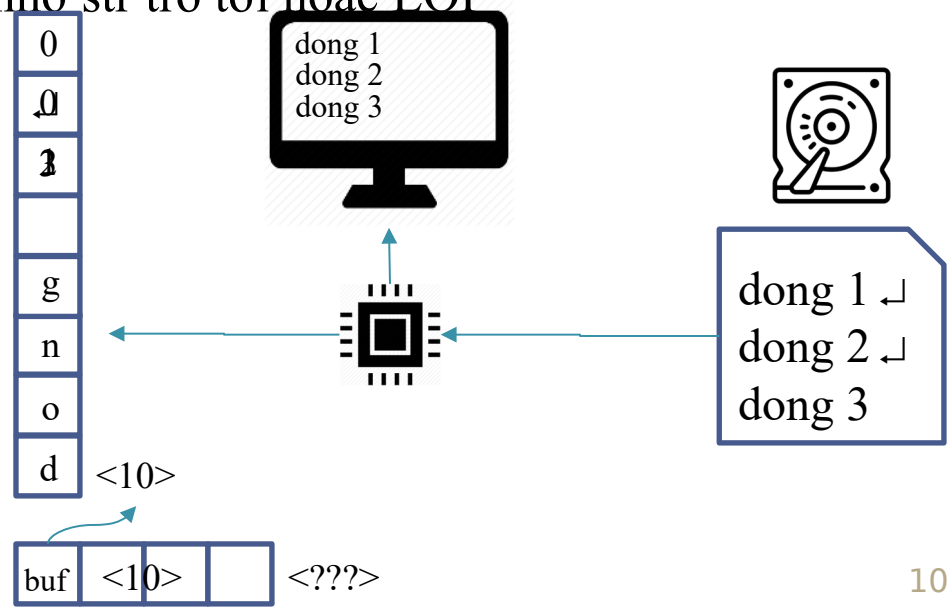


QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm đọc một chuỗi ký tự từ tập tin

  char* fgets(char* str, int n, FILE* fp)
- Ý nghĩa:
  - Đọc MỘT chuỗi ký tự từ tập tin fp vào vùng nhớ do str trỏ tới
  - Việc đọc kết thúc khi đủ n – 1 kí tự hoặc gặp kí tự '\n'
  - Lưu kí tự '\n' vào chuỗi nếu số kí tự đọc được < n – 1
  - Tự động thêm kí tự kết thúc chuỗi
- Giá trị trả về: địa chỉ vùng nhớ str trỏ tới hoặc EOF

```
1  FILE* fp = fopen("data.txt", "r");
2  char buf[8];
3  if(fp){
4    while(fgets(buf, 10, fp) != NULL){
5      printf("%s", buf);
6    }
7    fclose(fp);
8  }
```

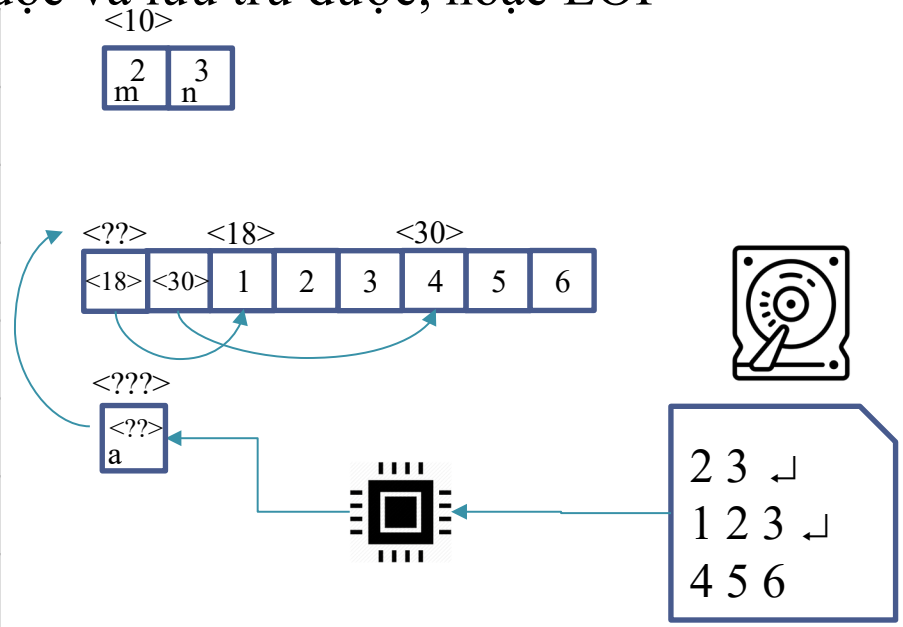


QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm đọc theo định dạng

  int fscanf(FILE* fp, char* fmt)
- Ý nghĩa:
  - Đọc dữ liệu theo định dạng fmt
  - Tương đương với scanf với fp ⇔ stdin
- Giá trị trả về: số thành phần đọc và lưu trữ được, hoặc EOF

```
0  int a[2][3] = {0}, m, n;
1  FILE* fp = fopen("data.txt", "r");
2  fscanf(fp, "%d", &m); fscanf(fp, "%d", &n);
3  if(fp){
4    for(int i = 0; i < m; i++)
5      for(int j = 0; j < n; j++)
6        fscanf(fp, "%d", &a[i][j]);
7    fclose(fp);
8  }
```

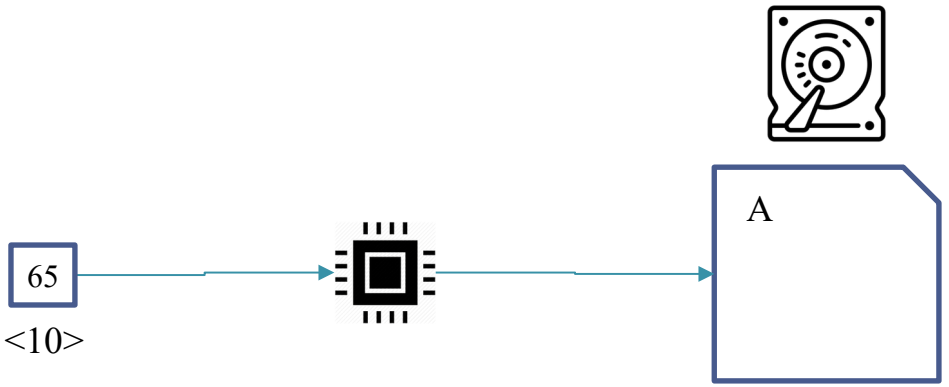


QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm ghi kí tự lên tập tin: int fputc(int ch, FILE* fp)
  - Ý nghĩa: ghi kí tự 'ch' vào tập tin fp
  - Giá trị trả về: trả về ký tự ch hoặc EOF
- Hàm ghi chuỗi kí tự lên tập tin

  int fputs(const char* str, FILE* fp)
  - Ý nghĩa: ghi chuỗi kí tự 'str' vào tập tin fp
  - Giá trị trả về: trả về kí tự cuối cùng đã ghi hoặc EOF

```
1  FILE* fp = fopen("data.ahihi", "w");
2  char ch = 65;
3  if(fp){
4    if(fputc(ch, fp) != EOF){
5      printf("Thanh cong");
6    }
7    fclose(fp);
8  }
```

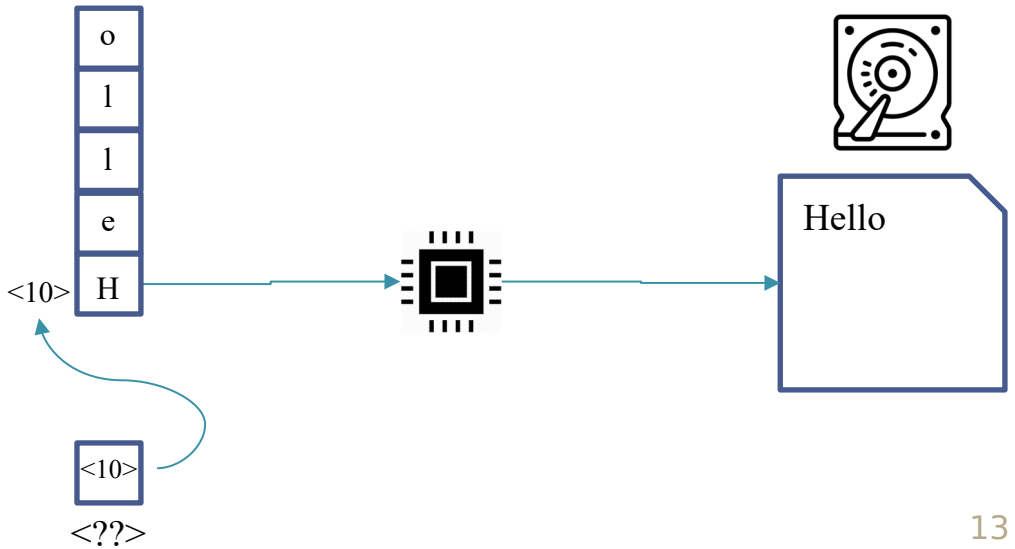


QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm ghi kí tự lên tập tin: int fputc(int ch, FILE* fp)
  - Ý nghĩa: ghi kí tự 'ch' vào tập tin fp
  - Giá trị trả về: trả về ký tự ch hoặc EOF
- Hàm ghi chuỗi kí tự lên tập tin

  int fputs(const char* str, FILE* fp)
  - Ý nghĩa: ghi chuỗi kí tự 'str' vào tập tin fp (không ghi '\0')
  - Giá trị trả về: trả về kí tự cuối cùng đã ghi hoặc EOF

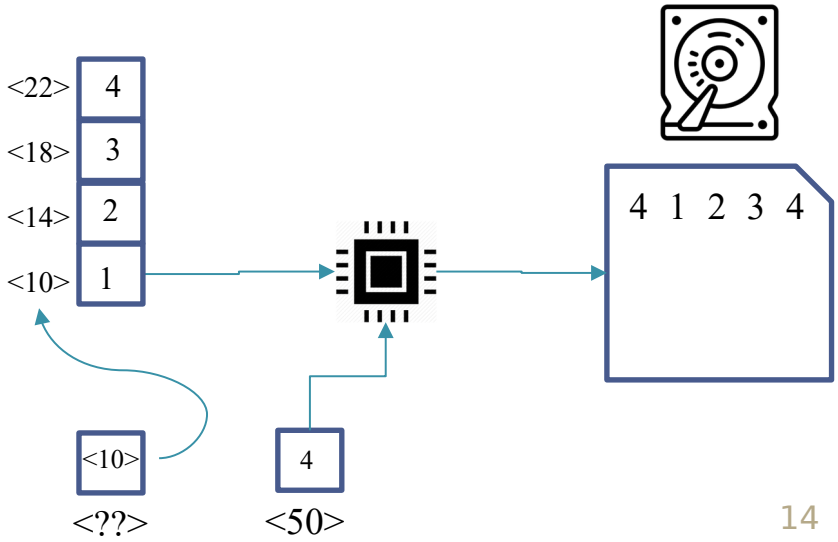
```
1  FILE* fp = fopen("data.ahihi", "w");
2  char ch[] = "Hello";
3  if(fp){
4    if(fputs(ch, fp) != EOF){
5      printf("Thanh cong");
6    }
7    fclose(fp);
8  }
```



QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm ghi theo định dạng: int fprintf(FILE* fp, char* fmt, …)
  - Ý nghĩa:
    - Ghi dữ liệu theo định dạng fmt vào tập tin fp.
    - Tương đương với printf với fp ⇔ stdout
  - Giá trị trả về: số byte ghi được hoặc EOF
- Hàm dọn dẹp vùng đệm: fflush(FILE* fp) hoặc int flushall();
  - Ý nghĩa: cập nhật thay đổi vào tập tin mà không cần đóng tập tin

```
1  int a[] = {1, 2, 3, 4}, n = sizeof(a)/sizeof(int);
2  FILE* fp = fopen("test.inp", "w");
3  if(fp){
4    fprintf(fp, "%d " , n);
5    for(int i = 0; i < n; i++)
6      fprintf(fp, "%d ", a[i]);
7    fclose(fp);
8  }
```



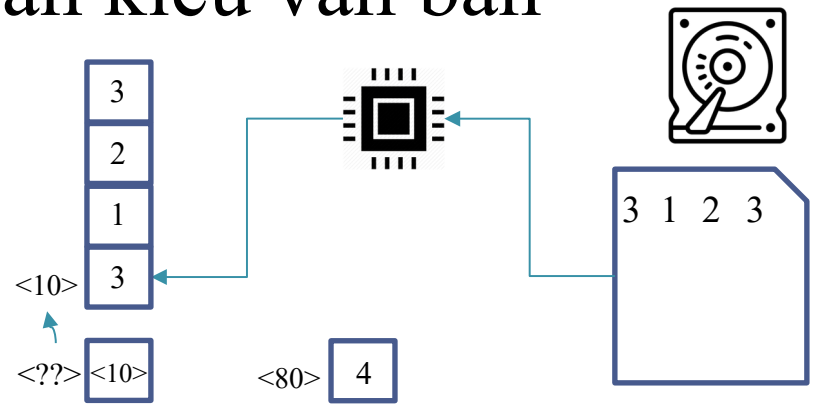
QUI TRÌNH LẬP TRÌNH TẬP TIN

- Các hàm liên quan tới tập tin nhị phân:
  - Đôi khi cần xử lý chính xác từng byte
  - Thay các tùy chọn ‘t’ thành ‘b’, ví dụ thay ‘rt’ thành ‘rb’, ‘wt’ thành ‘wb’
  - Có thể dời ‘vị trí đọc’ tới đúng nơi cần đọc/ghi
- Vị trí con trỏ FILE
  - Khi mở FILE để đọc/ghi (‘wb’ hay ‘rb’) thì con trỏ nằm ở đầu tập tin
  - Khi mở FILE để thêm (‘ab’) thì con trỏ nằm ở cuối tập tin

QUI TRÌNH LẬP TRÌNH TẬP TIN

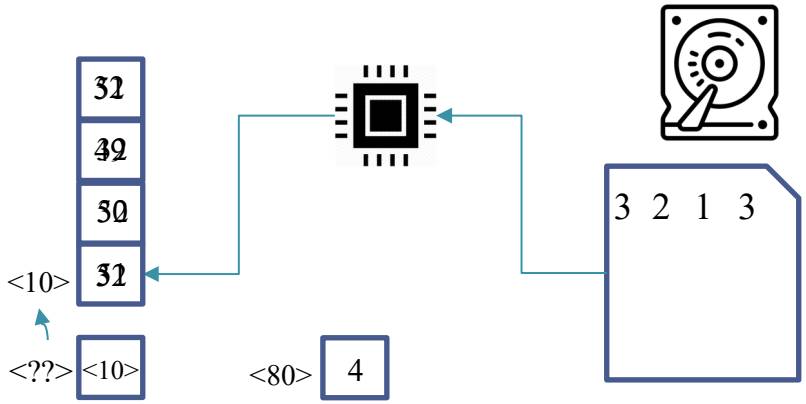
- Ví dụ đọc file văn bản kiểu văn bản

```
1  int b[4] = {0}, n = sizeof(b)/sizeof(int);
2  FILE* fp = fopen("test.inp", "r");
3  if(fp){
4    for(int i = 0; i < n; i++) fscanf(fp, "%d", &b[i]);
5    fclose(fp);
6  }
```



- Ví dụ đọc file văn bản kiểu nhị phân

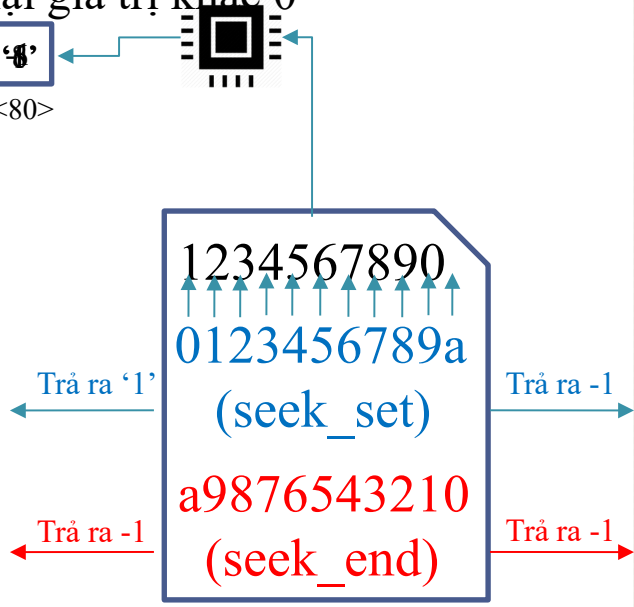
```
1  int b[4] = {0}, n = sizeof(b)/sizeof(int);
2  FILE* fp = fopen("test.inp", "rb");
3  if(fp){
4    for(int i = 0; i < n; i++) {
5      fread(&b[i], sizeof(char), 1, fp); // in b[i] ra xem
6      fread(&b[i], sizeof(char), 1, fp); // in b[i] ra xem
7    }
8   fclose(fp); }
```



QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm dời con trỏ định vị về đầu tập tin: void rewind(FILE* fp)
  - Ý nghĩa: dời vị trí định vị về đầu tập tin (byte 0)
  - Giá trị trả về: không có
- Hàm dời con trỏ định vị: int fseek(FILE* fp, long offset, int origin)
  - Ý nghĩa: đặt lại vị trí con trỏ theo offset so với cột mốc origin (SEEK_SET, SEEK_CUR, SEEK_END)
  - Giá trị trả về: trả về 0 nếu thành công, ngược lại giá trị khác 0

```
1  FILE* fp = fopen("test.inp", "rt");
2  if(fp){
3    fseek(fp, 5, SEEK_SET); printf("%c\n", fgetc(fp));
4    fseek(fp, -3, SEEK_END); printf("%c\n", fgetc(fp));
5    fseek(fp, 0L, SEEK_SET); printf("%c\n", fgetc(fp));
6    fseek(fp, 0L, SEEK_END); printf("%d\n", fgetc(fp));
7    rewind(fp); printf("%c\n", fgetc(fp));
8    fclose(fp);
9  }
```



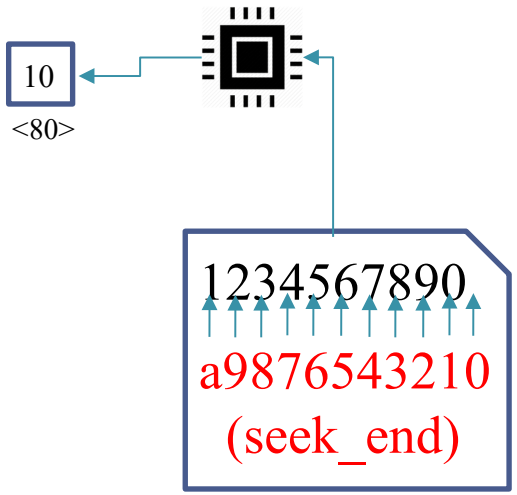
QUI TRÌNH LẬP TRÌNH TẬP TIN

- Hàm xác định vị trí con trỏ định vị

  long ftell(FILE* fp)

  - Ý nghĩa: trả ra vị trí hiện tại (tính từ 0)
  - Giá trị trả về: trả về vị trí hiện hành nếu thành công hoặc -1L nếu thất bại

```
1  FILE* fp = fopen("test.inp", "rb");
2  if(fp){
3    fseek(fp, 0L, SEEK_END);
4    long size = ftell(fp);
5    printf("%d\n", size);
6    fclose(fp);
7  }
```



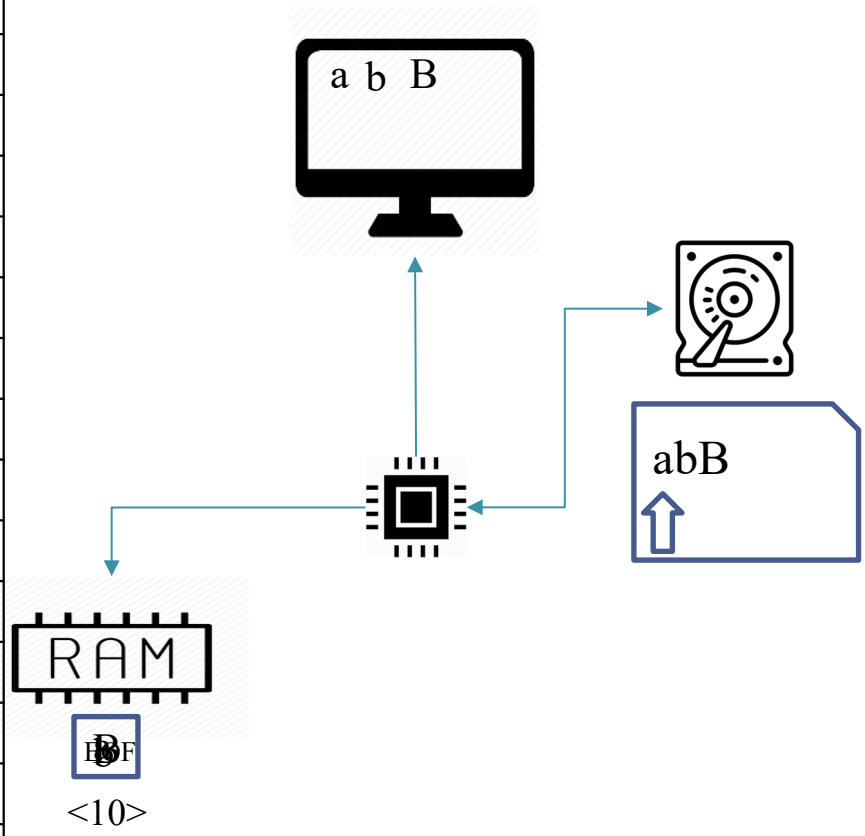
NỘI DUNG

- Giới thiệu
- Quy trình lập trình tập tin
- Các kĩ thuật hỗ trợ
- Ứng dụng

CÁC KĨ THUẬT HỖ TRỢ

- Ví dụ 1: đọc tập tin văn bản ASCII có sẵn

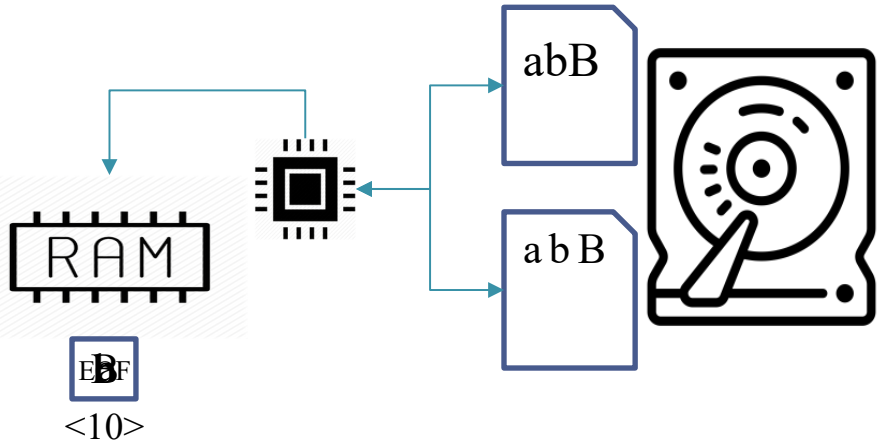
```
1  void transFile(FILE* inF, FILE* outF){
2    int ch;
3    while(1){
4      ch = fgetc(inF);
5      if(feof(inF) == false)
6        fputc(ch, outF);
7      else break;
8    }
9  }
10 void main(){
11   FILE* fp = fopen("Data.txt", "rt");
12   if(fp == NULL) return;
13   transFile(fp, stdout);
14   fclose(fp);
15 }
```



CÁC KĨ THUẬT HỖ TRỢ

- Ví dụ 2: sao chép nội dung tập tin

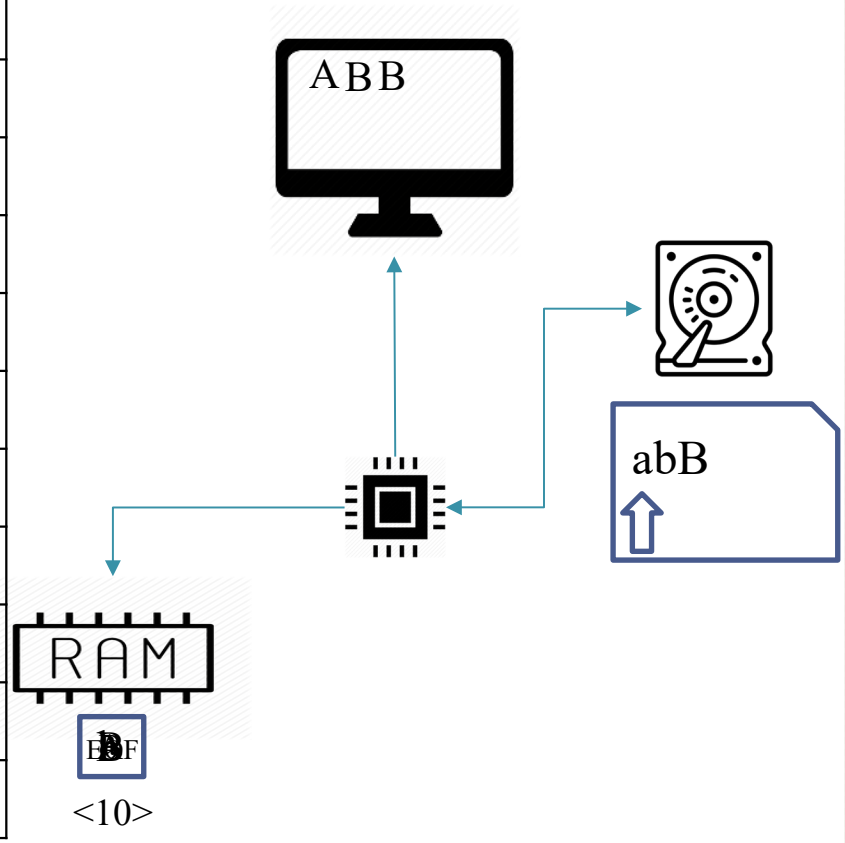
```
1  void main(){
2    FILE* fpIn, *fpOut;
3    fpIn = fopen("Data.txt", "rt");
4    if(fpIn == NULL) return;
5    fpOut = fopen("DataCopy.txt", "wt");
6    if(fpOut == NULL) {fclose(fpIn); return;}
7    transFile(fpIn, fpOut);
8    fclose(fpIn);
9    fclose(fpOut);
10 }
```



CÁC KĨ THUẬT HỖ TRỢ

- Ví dụ 3: ghi nội dung tập tin (có sửa đổi)

```
1  void convertFile(FILE* inFile, FILE* outFile){
2    int ch;
3    while(1){
4      ch = fgetc(inFile);
5      if(feof(inFile) == false){
6        ch = toupper(ch);
7        fputc(ch, outFile);
8      }
9      else break;
10   }
11 }
```

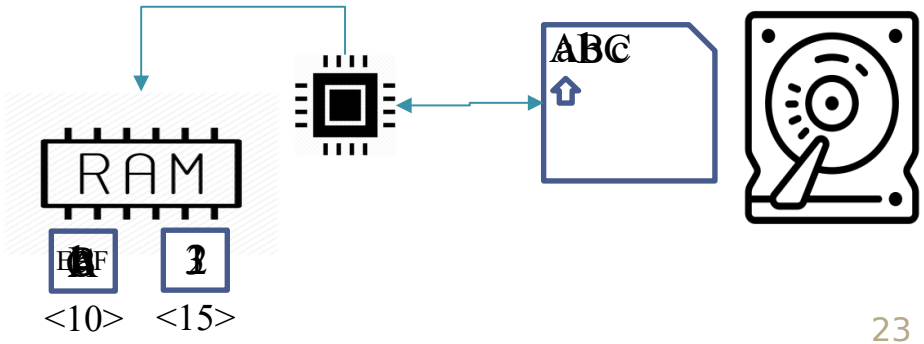


CÁC KĨ THUẬT HỖ TRỢ

- Ví dụ 4: sửa đổi trên một tập tin

Dòng	Mô tả
1	void UpcaseFile(FILE* f){
2	char ch; long pos;
3	while(1){
4	ch = fgetc(f);
5	if(feof(f)) break;
6	pos = ftell(f);
7	fseek(f, pos – 1, SEEK_SET);
8	ch = toupper(ch);
9	fputc(ch, f);
10	fseek(f, pos, SEEK_SET);
11	}
12	}

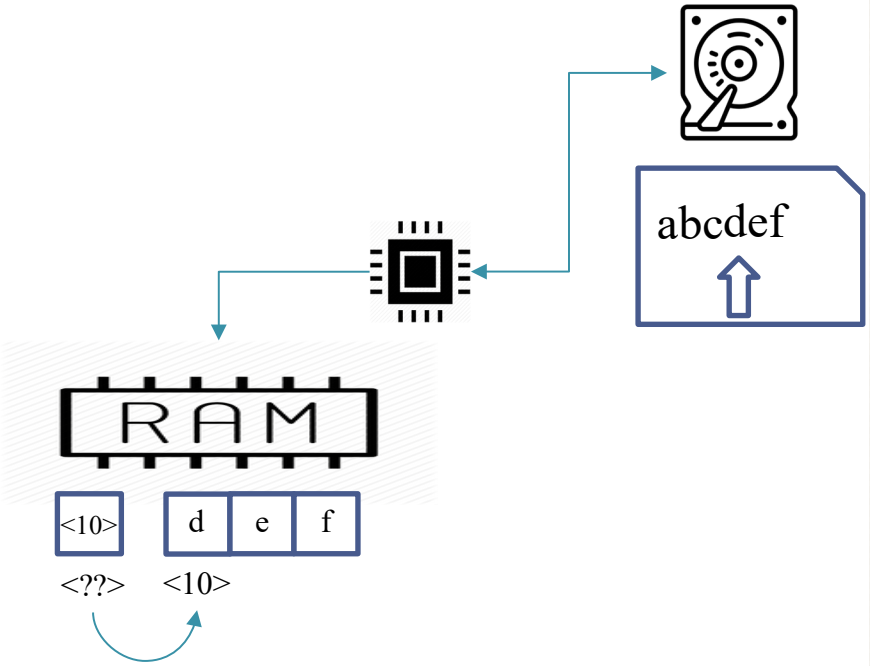
Dòng	Mô tả
13	void main(){
14	FILE* fp = fopen("Data.txt", "r+b")
15	if(fp == NULL) return;
16	UpcaseFile(fp);
17	fclose(fp);
18	}



CÁC KĨ THUẬT HỖ TRỢ

- Ví dụ 5: ghi thêm vào cuối tập tin

```
1  void writeMoreFile(FILE* f){
2    char buf[] = "def";
3    fputs(buf, f);
4  }
5  void main(){
6    FILE* fp = fopen("test.inp", "at");
7    if(fp == NULL) return;
8    writeMoreFile(fp);
9    fclose(fp);
10 }
```



CÁC KĨ THUẬT HỖ TRỢ

- Ví dụ 6: kiểm tra tập tin tồn tại + readonly

```
1  int isExist(char* fn){
2    FILE* f = fopen(fn, "rb");
3    if(fp == NULL) return 0;
4    fclose(f);
5    return 1;
6  }
7  int isReadonly(char* fn){
8    FILE* f = fopen(fn, "r+b");
9    if(fp == NULL) return 1;
10   fclose(fp);
11   return 0;
12 }
```

```
13 void checkFile(char* fn){
14   if(isExist(fn)){
15     printf("File %s is already exist\n", fn);
16     if(isReadonly(fn))
17       printf("File %s is readonly\n", fn);
18     else printf("File %s can be written and edited\n", fn);
19   }
20   else printf("File %s not found\n", fn);
21 }
22 void main(){
23   char* fname = "data.inp"; checkFile(fname);
24 }
```

CÁC KỸ THUẬT HỖ TRỢ

- Hàm xóa tập tin:

int remove(char* fname)

  - Ý nghĩa: trả ra 0 nếu thành công, -1 nếu thất bại

- Hàm đổi tên tập tin:

int rename(char* oldName, char* newName)

  - Ý nghĩa: trả ra 0 nếu thành công, -1 nếu thất bại

```
1  void main(){
2    char* oName = “datum.inp”, *nName = “data.inp”;
3    rename(oName, nName);
4    remove(nName);
5  }
```

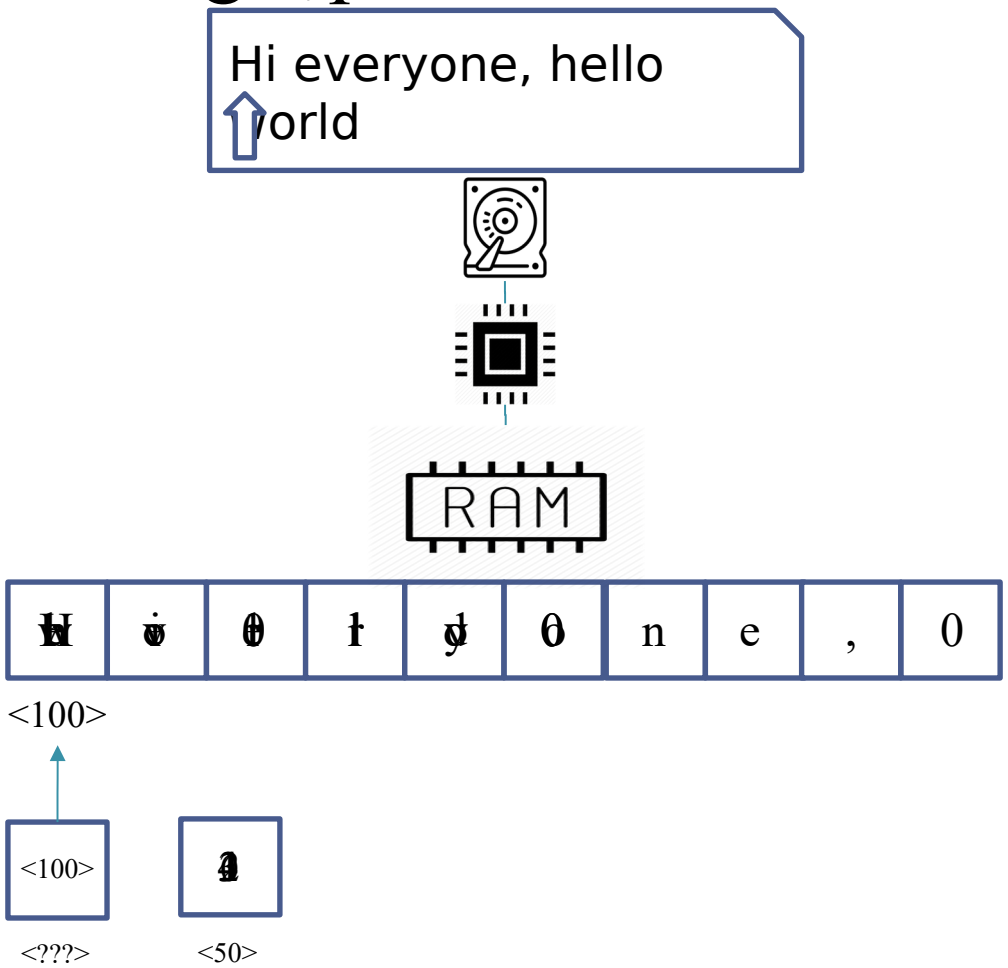
NỘI DUNG

- Giới thiệu
- Quy trình lập trình tập tin
- Các kĩ thuật hỗ trợ
- Ứng dụng

ỨNG DỤNG (KÝ TỰ & CHUỖI)

- ỨD1: đếm số từ trong tập tin ASCII

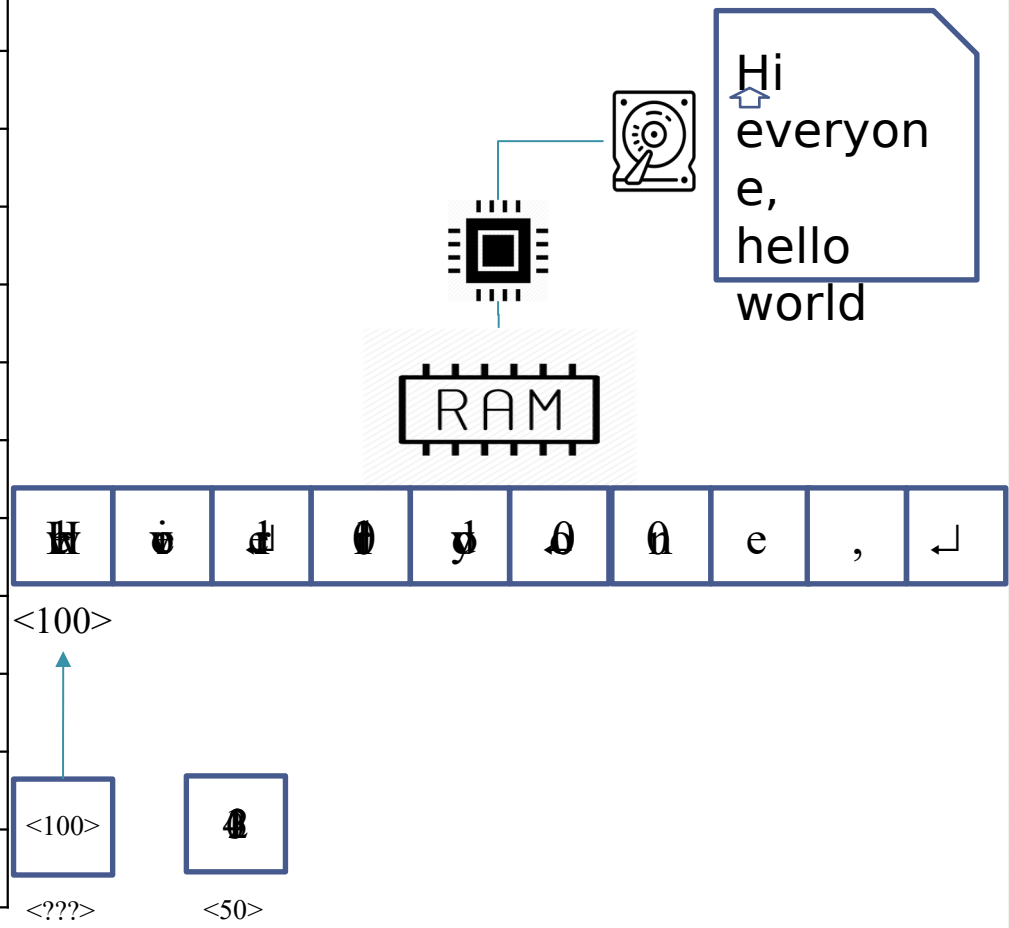
```
0  #define SIZE 10
1  long countWord(FILE* f){
2    long c = 0; char buf[SIZE];
3    while(fscanf(f, "%s", buf) > 0) c++;
4    return c;
5  }
6  void main(){
7    FILE* f = fopen("test.inp", "rt");
8    if(fp == NULL) return;
9    printf("%ld\n", countWord(fp));
10   fclose(fp);
11 }
```



ỨNG DỤNG (KÝ TỰ & CHUỖI)

- ỨD2: đếm số dòng trong tập tin ASCII

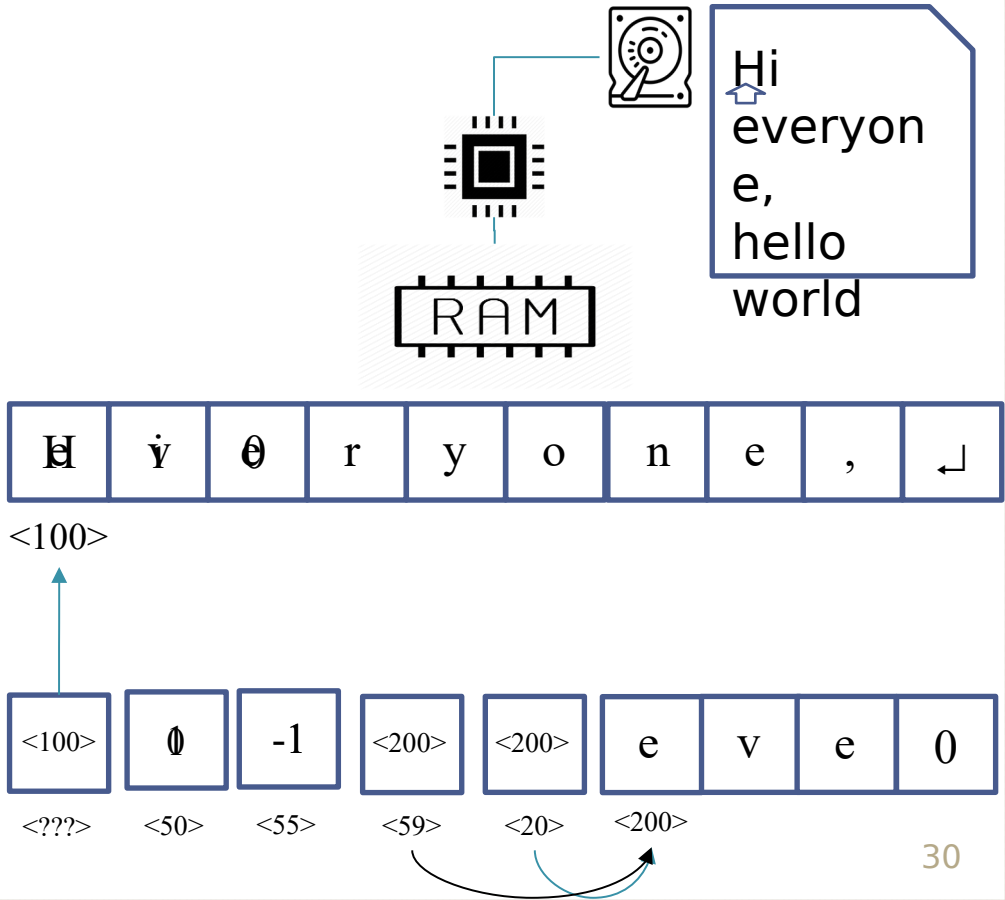
```
0  #define SIZE 10
1  long countLine(FILE* f){
2    long c = 0; char b[SIZE];
3    while(fgets(b, SIZE, f) != NULL) c++;
4    return c;
5  }
6  void main(){
7    FILE* f = fopen("test.inp", "rt");
8    if(fp == NULL) return;
9    printf("%ld\n", countLine(fp));
10   fclose(fp);
11 }
```



ỨNG DỤNG (KÝ TỰ & CHUỖI)

- ỨD3: Kiểm tra chuỗi "s" có khớp với phần đầu dòng nào trong tập tin ASCII

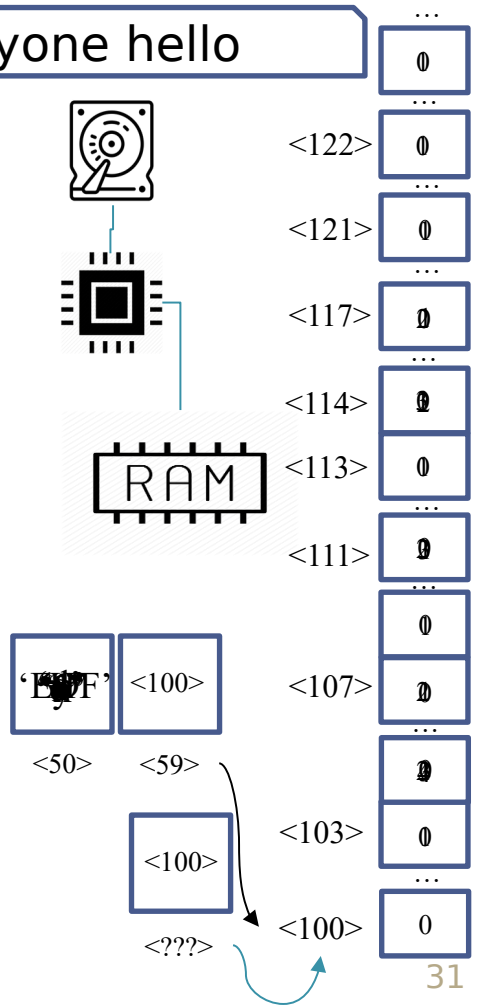
```
0  #define SIZE 10
1  long findLine(FILE* fi, char* s){
2    long c = 0, f = -1; char b[SIZE];
3    while(fgets(b, SIZE, fi) != NULL) {
4      if(strncmp(s, b, strlen(s)) == 0) {
5        f = c; break;
6      }
7      c++;
8    }
9    return f; }
10 void main(){
11   FILE* fp = fopen("test.inp", "rt");
12   if(fp == NULL) return;
13   char* str = "eve";
14   printf("%d\n", findLine(fp, str));
15   fclose(fp);
16 }
```



ỨNG DỤNG (KÝ TỰ & CHUỖI)

- ỨD4: Thống kê tần số xuất hiện các ký tự trong tập tin ASCII

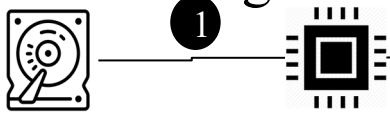
```
0  #define SIZE 'Z' – 'A' + 1 // 90 – 65 + 1 = 26
1  void countAlphabet(FILE* fi, long c[]){
2    for(int i = 0; i < SIZE; i++) c[i] = 0;
3    while(!feof(fi)){
4      char ch = fgetc(fi);
5      if(ch == EOF) break;
6      if('A' <= toupper(ch) && toupper(ch) <= 'Z') c[toupper(ch) – 'A']++;
7    }}
8  void showCount(FILE* fo, long c[], int n){
9    for(int i = 0; i < n; i++) fprintf(fo, "So luong %c la: %ld\n", 'A' + i, c[i]);}
10 void main(){
11   FILE* fp = fopen("test.inp", "rt"); long cnt[SIZE];
12   if(fp == NULL) return;
13   countAlphabet(fp, cnt);
14   showCount(stdout, cnt, SIZE);
15   fclose(fp);
16 }
```



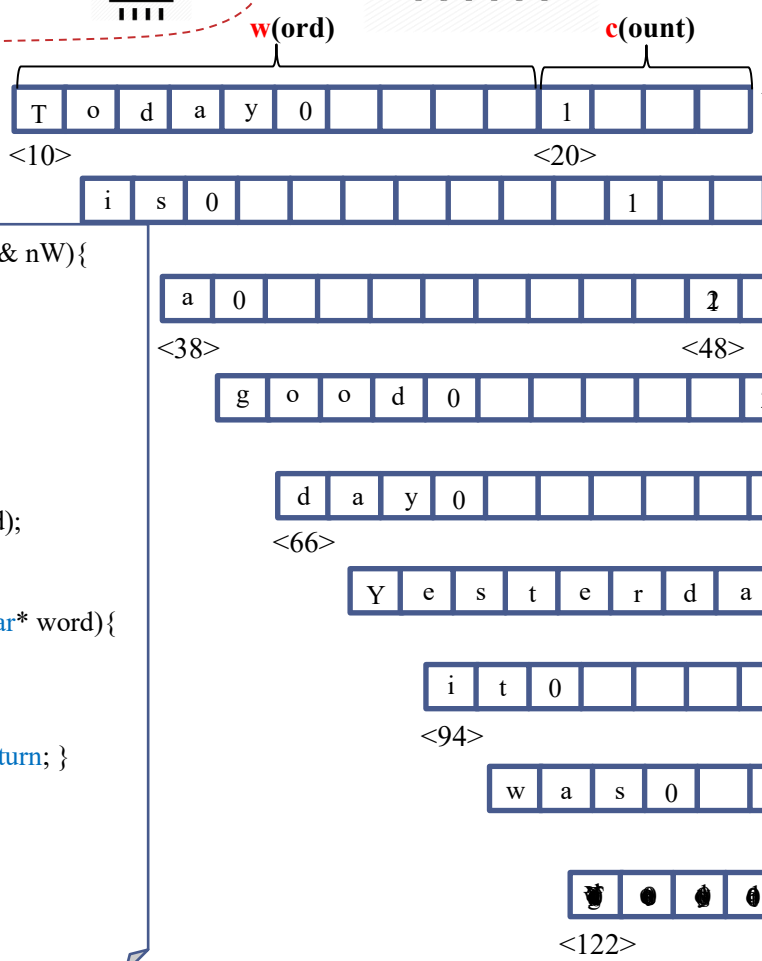
ỨNG DỤNG (KÝ TỰ & CHUỖI)

ỨD5: Thống kê tần số xuất hiện các từ trong tập tin ASCII

Today is a good day. Yesterday, it was a good day.




```
#define MaxW 11
#define wMaxLen 10
typedef struct {
  char w[wMaxLen];
  long c;
}WEntry;
```

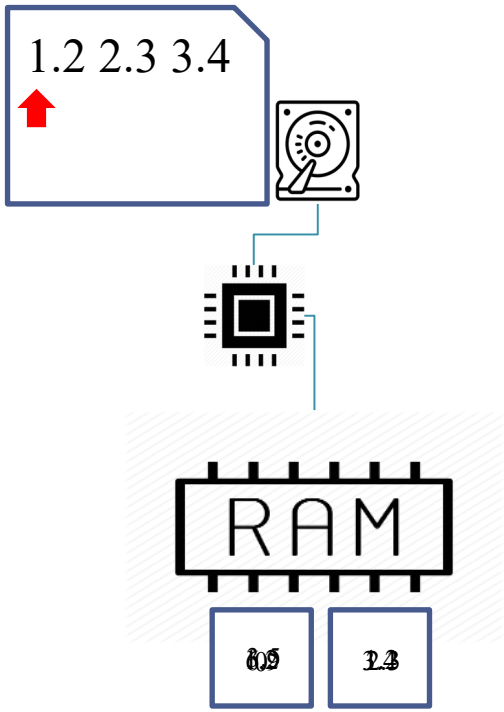



```
void countEachWord(FILE* fi, WEntry W[], int& nW){
  char word[wMaxLen] = ""; int ch, len = 0;
  while(!feof(fi)){
    ch = fgetc(fi);
    if(ch == EOF) break;
    ch = tolower(ch);
    if('a' <= ch && ch <= 'z') word[len++] = ch;
    else{
      word[len] = 0; checkNewWord(W, nW, word);
      word[0] = len = 0;
    }}}
void checkNewWord(WEntry W[], int& nW, char* word){
  if(!word || word[0] == 0) return;
  int n = nW;
  while(n--){
    if(strcmp(W[n].w, word) == 0){ W[n].c++; return; }
  }
  if(nW < MaxW){
    WEntry newWord;
    strcpy(newWord.w, word); newWord.c = 1;
    W[nW] = newWord; nW++;
  }}
```

ỨNG DỤNG (SỐ)

- ỨD1: Tính và in ra tổng các số thực trong tập tin

```
1  void fSum(FILE* f){
2    double sum = 0, num;
3    while(fscanf(f, "%lf", &num) > 0){//=1
4      sum += num;
5    }
6    return sum;
7  }
8  void main(){
9    FILE* fp = fopen("Data.txt", "rt");
10   if(fp == NULL) return;
11   printf("%lf\n", fSum(fp));
12   fclose(fp);}
```



ỨNG DỤNG (SỐ)

- ỨD2.1: in số lớn nhất từ tập tin số > 0

```
1  void fMax(FILE* f){
2    double max = 0, num;
3    while(fscanf(f, "%lf", &num) > 0){
4      if(max < num) max = num;
5    }
6    return max; }
```

1.2 2.3 3.4

- ỨD2.2: in số lớn nhất từ tập tin số thực

```
1  double fMax(FILE* f, int& err){
2    double max = 0, num;
3    err = 0;
4    while(fscanf(f, "%lf", &num) <= 0){
5      err = 1; return max;
6    }
7    max = num;
8    while(fscanf(f, "%lf", &num) > 0) {
9      if(max < num) max = num;
10   }
11   return max;
12 }
```

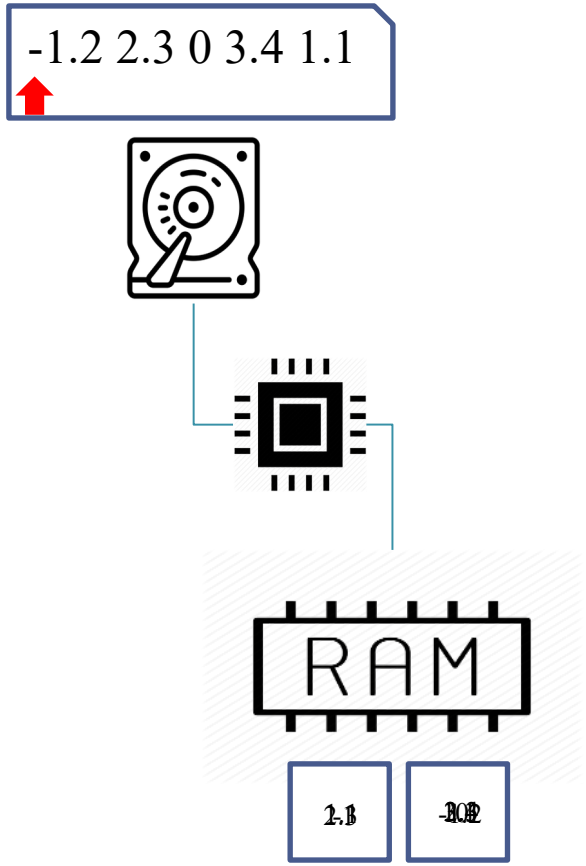
```
13 void main(){
14   FILE* fp = fopen("data.inp", "rt");
15   if(!fp) return;
16   int err; double m = fMax(f, error);
17   if(!err) printf("%lf\n", max);
18   fclose(fp);
19 }
```

-1.2 2.3 0 3.4

ỨNG DỤNG (SỐ)

- ỨD3: in số dương nhỏ nhất từ tập tin

```
1  double fMinPositive(FILE* f){
2    double Min = -1, n;
3    while(fscanf(f, "%lf", &n) > 0){
4      if(n > 0) {
5        if(Min == -1 || Min > n) Min = n;
6      }
7    }
8    return Min;
9  }
10 void main(){
11   FILE* fp = fopen("data.inp", "rt");
12   if(!fp) return;
13   double m = fMinPositive(fp);
14   if(m > 0) printf("%lf\n", m);
15   fclose(fp);
16 }
```



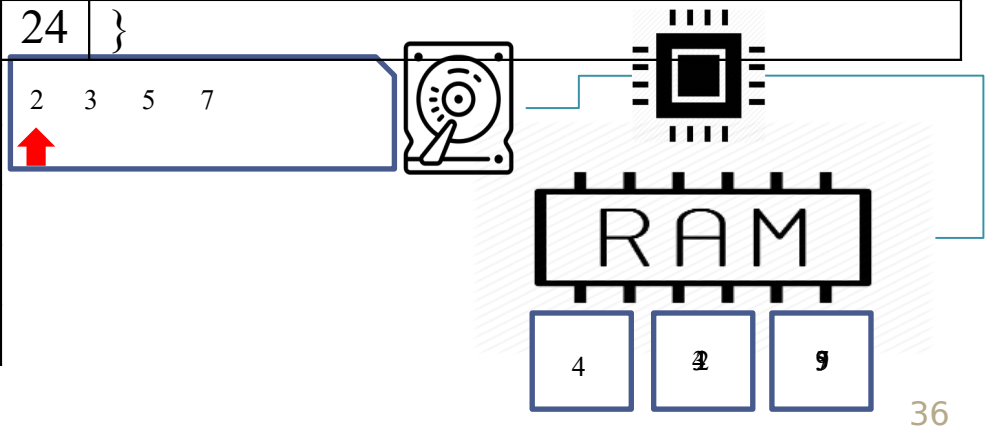
<59>

ỨNG DỤNG (SỐ)

- ỨD4: in n số nguyên tố lên tập tin

```
1  int isPrime(long n){ // … }
2  #define NUMBERLINE 20
3  void PrimesListing(int n, FILE* fo){
4    long p = 2, c = 1, nextn = 3;
5    fprintf(fo, "%ld ", p);
6    while(c < n){
7      if(isPrime(nextn)){
8        fprintf(fo, "%ld ", nextn);
9        c++;
10       if(c % NUMBERLINE == 0)
11         fprintf(fo, "\n");
12     }
13     nextn += 2;
14   }
15 }
```

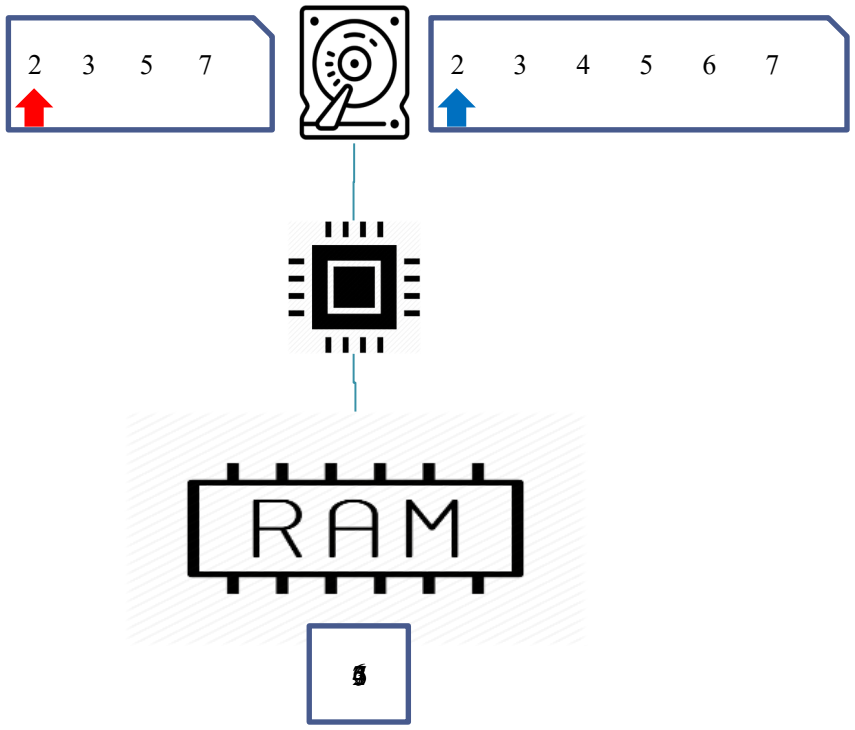
```
16 void main(){
17   int n; FILE* fp = fopen("prime.txt",
     "wt");
18   printf(" n = ");
19   scanf("%ld", &n);
20   if(fp != NULL)
21     PrimesListing(n, fp);
22     fclose(fp);
23   }
24 }
```



ỨNG DỤNG (SỐ)

- ỨD5: Lọc các số nguyên tố từ tập tin

```
1  int isPrime(long n){ // … }
2  void selectPrimes(FILE* fi, FILE* fo){
3    long p;
4    while(fscanf(fi, "%ld", &p) > 0){
5      if(isPrime(p))
6        fprintf(fo, "%ld ", p);
7    }
8  }
9  void main(){
10   FILE* fI = fopen("numbers.txt", "rt");
11   FILE* fO = fopen("primes.txt", "wt");
12   if(fI == NULL || fO == NULL) return;
13   selectPrimes(fI, fO);
14   fclose(fO);
15   fclose(fI);
16 }
```

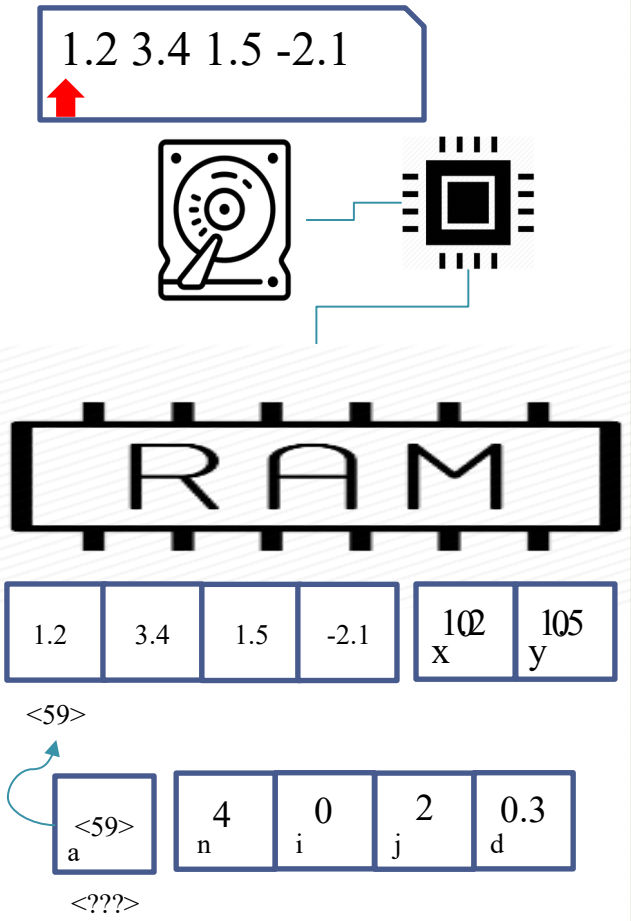


<59>

ỨNG DỤNG (SỐ)

- ỨD6: tìm hai số có khoảng cách nhỏ nhất trong tập tin các số thực

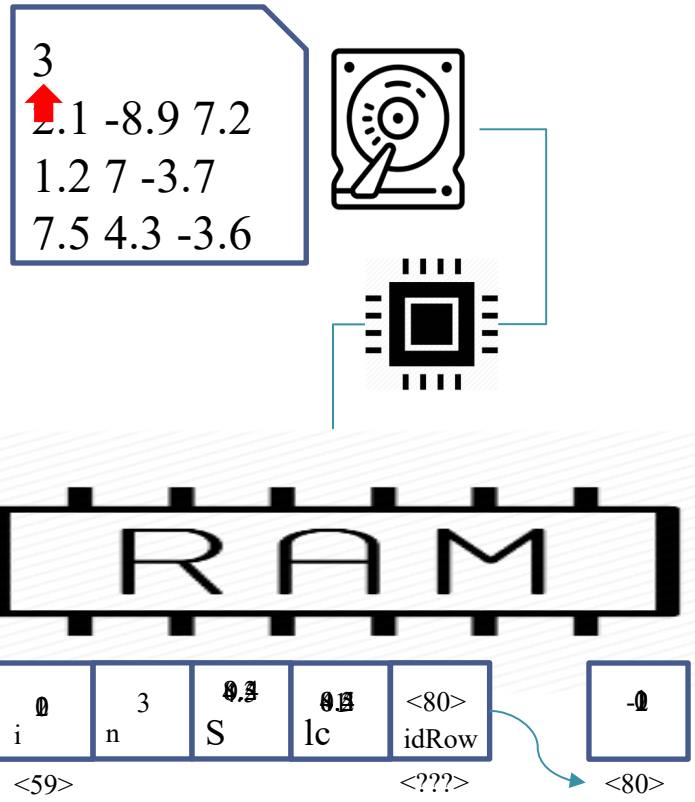
```
1  double minPair(double a[], int n, int &idi, int &idj){//…}
2  typedef SIZE 4; // chỉ xử lý trong RAM tối đa 4 phần tử
3  void getArray(FILE* f, double a[], int &n){
4      double x; n = 0;
5      while(fscanf(f, "%lf", &x) > 0) {
6        a[n] = x; n++;
7        if(n >= SIZE) break;
8      }
9  }
10 double findMinXY(FILE* f, double &x, double &y){
11   double a[SIZE], d; int n, i, j;
12   getArray(f, a, n);
13   x = y = 0; d = minPair(a, n, i, j);
14   if(d != -1) { x = a[i]; y = a[j]; }
15   return d;
16 }
```



ỨNG DỤNG (SỐ)

- ỨD7: tìm chỉ số dòng có tổng lớn nhất trong tập tin các số thực

```
1  double getCurRow(FILE* fi, int n){
2    double s = 0, tmp;
3    for(int i = 0; i < n; i++){
4      if(fscanf(fi, "%lf", &tmp) > 0) s += tmp;
5    return s;
6  }
7  double maxRow(FILE* fi, int &idRow){
8    double S, lc = -1; int n, i; idRow = -1
9    fscanf(fi, "%d", &n);
10   for(i = 0; i < n; i++){
11     S = getCurRow(fi, n);
12     if(idRow == -1 || lc < S){ lc = S; idRow = i; }
13   return lc;
14 }
```



ỨNG DỤNG (DỮ LIỆU PHỨC HỢP)

- Xét tập tin chứa thông tin các HOCSINH

1	0807621
2	7.5 10.0 9.5
3	0800682
4	8.9 8.3 7.0
5	0800455
6	9.1 6.5 8.0
7	0801516
8	6.8 9.0 8.5
9	0708334
10	8.5 5.5 8.0

Mã số: kiểu char[]
Điểm: kiểu float


```
struct pupil{
  char Code[8];
  double Gra1, Gra2, Gra3;
  double GPA;
};
typedef struct pupil PUPIL;
```

```
void getAll(FILE* f, PUPIL lst[], int& n){
  PUPIL p; n = 0;
  while(!feof(f)){
    if(n < 100 && getPupil(f, p)) { lst[n] = p; n++;}
    else break;
  }
}
int getPupil(FILE* f, PUPIL& p){
  fgetline(f, p.Code, 8);
  double s1, s2, s3;
  if(fscanf(f, "%lf %lf %lf", &s1, &s2, &s3)<=0) return 0;
  fgetc(f);
  p.Gra1 = s1; p.Gra2 = s2; p.Gra3 = s3; p.GPA = (s1 + s2 + s3)/3;
  return 1;
}
char* fgetline(FILE* f, char* str, int maxSize){
  if(feof(f)) return NULL;
  int ch, len = 0; str[len] = 0;
  do {
    ch = fgetc(f);
    if(ch == '\n' || ch == EOF) break;
    if(len < maxSIZE – 1) str[len++] = ch;
  }while(1);
  str[len] = 0;
  return str;
}
```