Nội dung

* Mối quan hệ giữa con trỏ và mảng
* Quản lý bộ nhớ cấp phát động

Mảng và con trỏ

* Tên mang = hằng con trỏ trở tới địa chỉ phần tử đầu tiên
* Có thể dùng con trỏ để truy cập các phần tử trong mảng
* + và - giá trị = dịch chuyển cách bao nhiêu phần tử so với phần tử hiện tại
* Mảng con trỏ

Int \*A; ==> con trỏ tới 1 biến nguyên

Int \*A[5] ==> Mảng A là mảng con trỏ 5 phần tử (mỗi phần tử sẽ là 1 con trỏ)

Ví dụ 1. muốn lưu trữ danh sách họ tên của 5 người

**Cách 1**. Dùng mảng tên với độ dài tên cố định (VD là 31 ký tự)

char HoTen[5][31];

* Độ dài các tên là cố định như nhau (31 ký tự, 30 + 1)

Nếu tên khác nhau độ dài?

Tối ưu bộ nhớ hơn? --> Dùng mảng con trỏ, mỗi con trỏ sẽ trỏ đến 1 tên

**Cách 2**. Dùng mảng con trỏ

Char \* HoTen2[5];

Mỗi phần tử sẽ được cấp phát động bộ nhớ riêng và có thể có kích thước riêng

HoTen2[0] = (char\*)malloc(sizeof(char)\*70);  
HoTen2[1] = (char\*)malloc(sizeof(char)\*10);

Nhận xét:

* Cách 1 không quan tâm tới giải phóng bộ nhớ
* Cách 1 không tối ưu bộ nhớ nếu độ dài các xâu này khác nhau nhiều (độ dài luôn bằng độ dài xâu tối đa)
* Cách 2. Tối ưu bộ nhớ hơn (độ dài mỗi phần tử nó tùy biến được)
* Cách 2. Phải giải phóng bộ nhớ mỗi khi dùng

**Con trỏ và cấp phát động bộ nhớ**

* Bộ nhớ cấp phát động được lấy từ HEAP --> có thể cấp phát cho các biến kích thước lớn
* Cấp phát thông thường 1 biến bị giới hạn kích thước là 2M (do trình biên dịch quy định) --> bạn KHÔNG thể lưu trữ biến kích thước lớn được

Gợi ý. Với các mảng >10 phần tử, nên dùng cấp phát động

Ví dụ 2. Đọc từ điển vào từ file và thực hiện bài toán tra cứu từ điển

* Từ điển khoảng 370100 từ
* Độ dài tối đa 1 từ khoảng 30 ký tự

Để cho đơn giản ta khai báo và cấp phát động 1 mảng Word[270100][30];

|  |
| --- |
| #include<stdlib.h>  #include<stdio.h>  #include<string.h>  const int TOTAL\_WORDS = 370100;  const int WORD\_LEN = 30;  int main()  {  char \*\*Words;  int i,n;  char filename[]="E:\\HUST\\C Basic\\20221\\Week 03 24-10-2022\\words\_alpha\\words\_alpha.txt";  FILE \*fPtr;  fPtr = fopen(filename, "r");  if(fPtr == NULL) {  perror("Error in opening file");  return(-1);  }  n = 0;  Words=(char\*\*)malloc(sizeof(char\*)\*TOTAL\_WORDS);  // cap phat do dai cac tu la co dinh  for(i=0;i<TOTAL\_WORDS; i++)  Words[i] = (char\*)malloc(sizeof(char)\*WORD\_LEN);    i=0;  while(fgets(Words[i], WORD\_LEN, fPtr)!=NULL) {  puts(Words[i]);  i++;  }  fclose(fPtr);  n=i;  printf("Total Read: %d\n",n);  // giai phong bo nho  for(i=0;i<TOTAL\_WORDS; i++)  free(Words[i]);    free(Words);    return 0;  } |

Nhược điểm của đoạn chương trình trên là kích thước các từ giống nhau là 30 ký tự --> KHÔNG tối ưu bộ nhớ.

Sửa lại để tối ưu bộ nhớ hơn?

|  |
| --- |
| #include<stdlib.h>  #include<stdio.h>  #include<string.h>  const int TOTAL\_WORDS = 370100;  const int WORD\_LEN = 30;  int main()  {  char \*\*Words;  int i,n,total\_size=0;;  char filename[]="E:\\HUST\\C Basic\\20221\\Week 03 24-10-2022\\words\_alpha\\words\_alpha.txt";  char buff[50];  FILE \*fPtr;  fPtr = fopen(filename, "r");  if(fPtr == NULL) {  perror("Error in opening file");  return(-1);  }  n = 0;  // cap phat mang con tro  Words=(char\*\*)malloc(sizeof(char\*)\*TOTAL\_WORDS);  i=0;  while(fgets(buff, WORD\_LEN, fPtr)!=NULL) {  Words[i] = (char\*)malloc(sizeof(char)\*(strlen(buff)+1));  strcpy(Words[i],buff);  puts(Words[i]);  total\_size=total\_size+strlen(buff)+1;  i++;  }  fclose(fPtr);  n=i;  printf("Total Read: %d\n",n);  printf("Total size: %d\n",total\_size);  // giai phong bo nho  for(i=0;i<TOTAL\_WORDS; i++)  free(Words[i]);    free(Words);    return 0;  } |

**Tối ưu được gần 7MB bộ nhớ!!!**

TRa cứu từ điển

* Đơn giản = Tuần tự
* Nâng cao hơn = Tìm kiếm nhị phân

|  |
| --- |
| #include<stdlib.h>  #include<stdio.h>  #include<string.h>  const int TOTAL\_WORDS = 370100;  const int WORD\_LEN = 30;  void standalizeWord(char \*Word)  {  int i=0;  while(i<strlen(Word))  {  if(Word[i]=='\r' || Word[i]=='\n')  {  Word[i]='\0';  break;  }  else  i++;  }  }  char \*\*loadDict(char\* filename, int\* dict\_size)  {  char \*\*Words;  int i,total\_size=0;;  char buff[50];  FILE \*fPtr;  fPtr = fopen(filename, "r");  if(fPtr == NULL) {  perror("Error in opening file");  return(NULL);  }  // cap phat mang con tro  Words=(char\*\*)malloc(sizeof(char\*)\*TOTAL\_WORDS);  i=0;  while(fgets(buff, WORD\_LEN, fPtr)!=NULL) {  standalizeWord(buff);  Words[i] = (char\*)malloc(sizeof(char)\*(strlen(buff)+1));  strcpy(Words[i],buff);  puts(Words[i]);  total\_size=total\_size+strlen(buff)+1;  i++;  }  fclose(fPtr);  printf("Total Read: %d\n",i);  printf("Total size: %d\n",total\_size);  \*dict\_size = i;  return Words;  }    void freeMem(char \*\*pointer)  {  // giai phong bo nho  for(int i=0;i<TOTAL\_WORDS; i++)  free(pointer[i]);    free(pointer);  }  int checkWord(char \*word, char \*\*Dict, int dict\_size)  {  for(int i=0; i<dict\_size; i++)  if(strcmp(word,Dict[i])==0) return 1;  return 0;  }  int main()  {  char \*\*Dict;  char lookupW[30];  int dict\_size;  char filename[]="E:\\HUST\\C Basic\\20221\\Week 03 24-10-2022\\words\_alpha\\words\_alpha.txt";  Dict = loadDict(filename, &dict\_size);  printf("Tra cuu tu dien:\n");  while(1)  {  printf("Tu can tra: ");  fflush(stdin);  gets(lookupW);  if(strcmp(lookupW,"@")==0) break;  if(checkWord(lookupW,Dict,dict\_size))  printf("Tu nay dung!\n");  else  printf("Tu nay KHONG co trong tu dien!\n");  }  freeMem(Dict);  return 0;  } |

**Bài tập 1**. Viết chương trình đọc vào **1 file văn bản** tiếng Anh. In ra màn hình những tên riêng (chứa chữ cái viết hoa) trong văn bản đó

VD văn bản đầu vào

|  |
| --- |
| Boris Johnson pulls out of race to be leader of UK’s Conservative Party and next prime minister |
| In ra:  Boris, Johnson, UK’s, Conservative, Party |

Làm sao tách được từ?

Từ ngăn cách bởi

* Dấu câu: . , ! ?
* Dấu cách trống

Cách đơn giản:

* Tiền xử lý biến các dấu câu thành dấu cách trống --> chỉ cần xử lý mỗi dấu cách trống làm dấu ngăn cách

Làm sao đọc được văn bản từ text file?

* Đọc từng dòng? Dùng fgets
* Đọc từng ký từ? Fgetchar
* Đọc từng đoạn với fscanf

Nên dùng cấp phát bộ nhớ động nếu có thể!

**Bài tập 2**. thông tin danh bạ điện thoại được lưu trong file text dạng

|  |
| --- |
| 3  Nguyen Van Anh  1:09123454  2:03456565  Tran Van Duong  1:09343434  Le Hoang Dung  1:03454232 |

Trong đó 3 là số lượng contact  
Mỗi contact sẽ có

* Tên
* SDT, nếu có nhiều hơn 1 sdt thì sẽ lần lượt là 1: 2: và 3:
* Mỗi thông tin sẽ được lưu trữ trên 1 dòng

Hãy viết chương trình

* Đọc vào danh bạ trên
* Bộ nhớ lưu trữ được cấp phát động
* Chức năng đơn giản nhất là tra cứu danh bạ theo từ khóa  
  VD. Tìm với “Du” sẽ trả về Tran Van Duong và Le Hoang Dung