ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

C Programming Basic

Basis - part 2

Content

- Review struct
- Dynamic allocation
- Binary files
- Exercises

Dynamic allocation

- Allocate memory at runtime instead of fixed-size allocate memory at compile time
- User pointers to manage allocated memory

malloc

void * malloc(unsigned int nbytes);

- Allocate a block of memory having **nBytes** size
- **malloc** returns a pointers to the allocated block if SUCCESS, and returns **NULL** if FAIL.
- Library **stdlib.h: #include <stdlib.h>**

Exercise

```c
#include <stdlib.h>
int main(){
    int i, n, *p;
    printf("How many numbers do you want to enter?\n");
    scanf("%d", &n);
    p = (int *)malloc(n * sizeof(int));
    if (p == NULL){
        printf("Memory allocation failed!\n");
        return 1;
    }
    /* input n integers */
    ...
    /* Display in a reversed order */
    ...

    free(p); /* deallocate */
    return 0;
}
```

Exercise

```
int main(){
  . . .
    /* Input n integers*/
    printf("Please enter numbers now:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &p[i]);

    /* Display n integers in a revere order*/
    printf("The numbers in reverse order are - \n");
    for (i = n - 1; i >= 0; --i)
        printf("%d ",p[i]);
    printf("\n");
    free(p);
    return 0;
}
```

Type Casting

- **void * malloc(unsigned int nbytes);**
- (**void ***) is a general pointer, and can be casted to any type.

- **p = (int *)malloc(n*sizeof(int));**

calloc

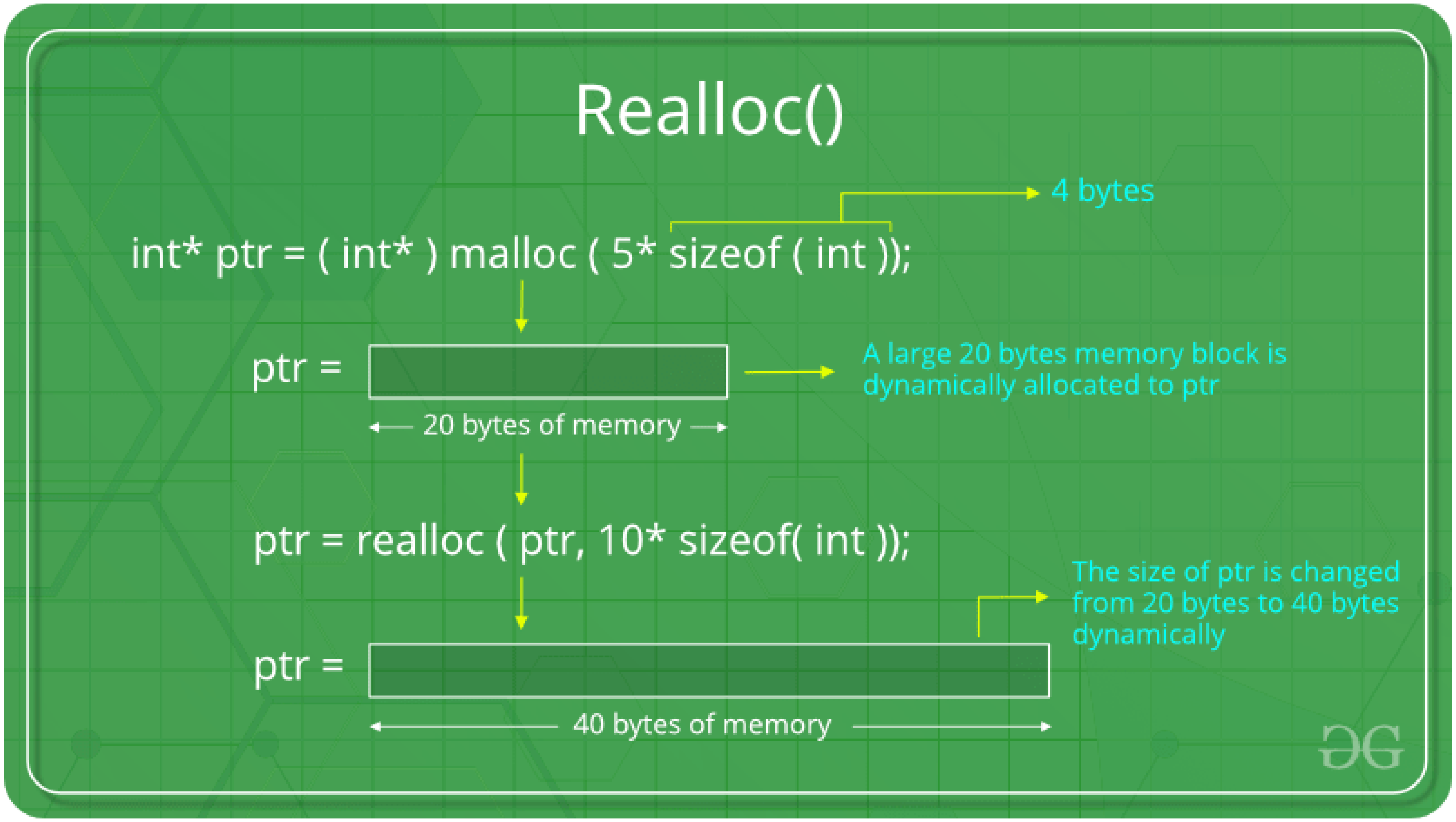
void *calloc(size_t nitems, size_t sz);

- Allocate a block of memory consisting of nitems of the same type, each item has the size of **sz** bytes
- Initialize value 0 for all items
- **ptr = (float*) calloc(25,sizeof(float));**

Reallocate

- Sometimes, we need to reallocate memory
- **void *realloc(void *ptr, size_t sz)**
    - Change the size of the block pointed by **ptr** which was allocated before
    - **sz**: is the size of the new block of memory

realloc


Realloc()
4 bytes
int* ptr = (int*) malloc (5* sizeof (int));
ptr =
A large 20 bytes memory block is dynamically allocated to ptr
20 bytes of memory
ptr = realloc (ptr, 10* sizeof(int));
The size of ptr is changed from 20 bytes to 40 bytes dynamically
ptr =
40 bytes of memory

Nguồn: geeksforgeeks

Example

```
#include <stdio.h>

#include <stdlib.h>

int main(){

   char *str;

   str = (char *) malloc(15);

   strcpy(str, "tutorialspoint");

   printf("String = %s,  Address = %u\n", str, str);

   /* reallocate*/

   str = (char *) realloc(str, 25);

   strcat(str, ".com");

   printf("String = %s,  Address = %u\n", str, str);

   free(str);

   return 0;

}
```

Deallocate

```
void free(void *ptr);
```

- Function **free(p)** deallocates the memory pointed by **p**
- If **p** is NULL, runtime exception will be raised

Exercise

- Implement the function **my_strcat** :
  - Input consists of two strings **s1** and **s2**
  - Output: pointer that points to the reallocated memory with the content which is the concatenation of s1 and s2
  - Example: The concatenation of “hello_” and “world!” return “hello_world!”
- Use dynamic reallocation
- Test your function

Implement: my_strcat

```c
char *my_strcat(char *str1, char *str2){
    int len1, len2; char *result;
    len1 = strlen(str1); len2 = strlen(str2);
    result = (char*)malloc((len1 + len2 + 1) *  sizeof(char));
    if (result == NULL) {
        printf("Allocation failed! Check memory\n");
        return NULL;
    }
    strcpy(result, str1);
    strcpy(result + len1, str2);
    return result;
}
```

Implement: hàm main()

```
int main(){
    char str1[MAX_LEN + 1], str2[MAX_LEN + 1];
    char *cat_str;
    printf("Please enter two strings\n");
    scanf("%100s", str1);
    scanf("%100s", str2);
    cat_str = my_strcat(str1, str2);
    if (cat_str == NULL)    {
        printf("Problem allocating memory!n");
        return 1;
    }
    printf("The concatenation of %s and %s is %s\n", str1, str2, cat_str);
    free(cat_str);
    return 0;
}
```

Homework

- Implement the function
  - char* subStr(char* s1, int offset, int number)
- Split the string s1 started from the index **offset** (the string indices start from 0) and has length of **number** characters.

Struct

- Group data items of different kinds under a name

```
struct struct-name{

    field-type1 field-name1;
    field-type2 field-name2;
    field-type3 field-name3;

            ...
};
```

Example: Complex type definition

```
struct complex {
      int real;
      int img;
  };
struct complex num1, num2, num3;
```

Hoặc

```
typedef struct complex {
      int real;
      int img;
} complex_t;


complex_t num1, num2;
```

Exercise

- Given two structs

```
typedef struct point{
    double x;
    double y;
} point_t;

typedef struct circle{
    point_t center;
    double  radius;
} circle_t;
```

- Write the function is_in_circle that returns 1 if point p is in the circle c.

Hint

```c
int is_in_circle(point_t *p, circle_t *c){
    double x_dist, y_dist;
    x_dist = p->x - c->center.x;
    y_dist = p->y - c->center.y;
    return (x_dist * x_dist + y_dist * y_dist <= c->radius * c->radius);
}
int main(){
    point_t  p;
    circle_t c;
    printf("Enter point coordinates\n"); scanf("%lf%lf", &p.x, &p.y);
    printf("Enter circle center coordinates\n");
    scanf("%lf%lf", &c.center.x, &c.center.y);
    printf("Enter circle radius\n"); scanf("%lf", &c.radius);
    if (is_in_circle(&p, &c))
        printf("point is in circle\n");
    else
        printf("point is out of circle\n");
    return 0;
}
```

Homework

- Write a program that check the intersection of two circles, use dynamic allocation.

Binary files

mode	Ý nghĩa
"rb"	Mở tập tin nhị phân đã có chỉ để đọc.
"wb"	Mở tập tin nhị phân chỉ để ghi.
"ab"	Mở tập tin nhị phân đã có để ghi thêm vào cuối.
"r+b"	Mở tập tin nhị phân đã có cho phép cả đọc và ghi.
"w+b"	Mở tập tin nhị phân cho phép cả đọc và ghi. Nếu file đã tồn tại, nội dung sẽ bị ghi đè. Nếu file không tồn tại, nó sẽ được tạo tự động.
"a+b"	Mở hoặc tạo tập tin nhị phân cho phép cả đọc và ghi vào cuối.

Input/output with data blocks

- C provides two I/O functions: fread() and fwrite(), allow users to manipulate I/O operations with blocks

fread()

- Function fread

  size_t fread(void *ptr, size_t sz, size_t n, FILE *stream);

- ptr is a pointer to an array used to store the data read
- sz: size of each element of the array (in bytes).
- n: number of elements loaded from the file
- stream: file pointer
- Return the number of element that are actually loaded from the file.

fwrite()

- Function fwrite

  size_t fwrite(const void *ptr, size_t sz, size_t n, FILE *stream);
- ptr is a pointer to the array that stores that blocks that will be written to the files
- **sz**: size of each element (in bytes).
- n: number of elements that will be written to the file.
- stream: file poitner
- Return the number of elements that are actually written to the file

Function feof

- Function int feof(FILE *stream);
- Check if the file pointer is at the end of the file
  - Return 0 if the file pointer is at the end of the file; and return positive integer value, oherwise

Example

- Read 80 byte from the file haiku.txt.

```
enum {MAX_LEN = 80};
int num;
FILE *fptr2;
char filename2[]= "haiku.txt";
char buff[MAX_LEN + 1];
if ((fptr2 = fopen(filename2, "r")) == NULL){
    printf("Cannot open %s.\n", filename2);
    reval = FAIL; exit(1);
}
. . . .
num = fread(buff, sizeof(char), MAX_LEN, fptr2);
buff[num * sizeof(char)] = `\0';
printf("%s", buff);
```

Exercise

- Write a program that copies the data (from lab1.txt to lab1a.txt) using fread and fwrite, feof

Hint

```
#include <stdio.h>
enum {SUCCESS, FAIL, MAX_LEN = 80};
void BlockReadWrite(FILE *fin, FILE *fout);
int main() {
         FILE *fptr1, *fptr2;
         char filename1[]= "lab1a.txt";
         char filename2[]= "lab1.txt";
         int reval = SUCCESS;
         if ((fptr1 = fopen(filename1, "w")) == NULL){
                  printf("Cannot open %s.\n", filename1);
                  reval = FAIL;
         } else if ((fptr2 = fopen(filename2, "r")) == NULL){
               printf("Cannot open %s.\n", filename2);  reval = FAIL;
         } else {
               BlocReadWrite(fptr2, fptr1);
               fclose(fptr1);   fclose(fptr2);
         }
         return reval;
}
```

Hint

```
void BlockReadWrite(FILE *fin, FILE *fout) {
         int num;
         char buff[MAX_LEN + 1];

         while (!feof(fin)){
          num = fread(buff, sizeof(char), MAX_LEN, fin);
          buff[num * sizeof(char)] = `\0';

          printf("%s", buff);
          fwrite(buff, sizeof(char), num, fout);
         }
}
```

Exercise

- Write a program mycat that performs the command **cat** in Unix using I/O based on blocks
- Hint:
  - Use command parameters
  - Use the function fread

Hint

```
#include <stdio.h>
enum {SUCCESS, FAIL, MAX_LEN = 80};
void BlockCat(FILE *fin);
main(int argc, char* argv[]) {
        FILE *fptr1, *fptr2;
        int reval = SUCCESS;
        if (argc !=2){
          printf("The correct syntax should be: cat1 filename \n");
          reval = FAIL;
        }
        if ((fptr1 = fopen(argv[1], "r")) == NULL){
                printf("Cannot open %s.\n", argv[1]);
                reval = FAIL;
        } else {
            BlocCat(fptr1);
            fclose(fptr1);
        }
        return reval;
}
```

Hint

```
void BlockCat(FILE *fin) {
         int num;
         char buff[MAX_LEN + 1];

         while (!feof(fin)){
          num = fread(buff, sizeof(char), MAX_LEN, fin);
          buff[num * sizeof(char)] = `\0';

          printf("%s", buff);
         }
}
```

Homework

- Write a program that copies the data from one file to another file with different options
    1. Copy by character
    2. Copy by line
    3. Copy by block - optional size
    4. Quit
- After finishing the copy operation, print the execution time to compare the performance between the options
  Note: Source file is a text file with at least 640KB.

Exercise

- Write a program that manipulates with phone books. Each element contains: "name," "telephone number," "e-mail address”.
- Declare an array for storing the phone book in the memory
- Read 10 elements from stdin
- Write the array to a file phonebook.dat using fwrite.
- Load the phone book data from the file phonebook.dat into an array in the memory, print out the information to the screen for checking the correctness of the above operations

Hint

```
enum {SUCCESS, FAIL, MAX_ELEMENT = 20};


// the phone book structure
typedef struct phoneaddress_t {
      char name[20];
      char tel[11];
      char email[25];
}phoneaddress;
```

Hint

```
#include <stdio.h>
enum {SUCCESS, FAIL, MAX_ELEMENT = 20};
// the phone book structure
typedef struct phoneaddress_t {
      char name[20];
      char tel[11];
      char email[25];
}phoneaddress;

int main(void)
{
   FILE *fp;
   phoneaddress phonearr[MAX_ELEMENT];
     int i,n, irc; // return code
   int reval = SUCCESS;
```

Hint

```
 printf("How many contacts do you want to enter (<20)?");
scanf("%d", &n);
for (i=0; i<n; i++){
  printf("name:"); scanf("%s",phonearr[i].name);
  printf("tel:"); scanf("%s",phonearr[i].tel);
  printf("email:"); scanf("%s",phonearr[i].email);
}
 if ((fp = fopen("phonebook.dat","w+b")) == NULL){
      printf("Can not open %s.\n", "phonebook.dat");
      reval = FAIL;
 }
// write the entire array into the file
irc = fwrite(phonearr, sizeof(phoneaddress), n, fp);
printf(" fwrite return code = %d\n", irc);
fclose(fp);
```

Hint

```
    //read from this file to array again
   if ((fp = fopen("phonebook.dat","rb")) == NULL){
         printf("Can not open %s.\n", "phonebook.dat");
         reval = FAIL;
    }
   irc = fread(phonearr, sizeof(phoneaddress), n, fp);
   printf(" fread return code = %d\n", irc);
   for (i=0; i<n; i++){
     printf("%s-",phonearr[i].name);
     printf("%s-",phonearr[i].tel);
     printf("%s\n",phonearr[i].email);
   }
   fclose(fp);
   return reval;
}
```

Homework

- Write a program that read the grade information from a text file bangdiem.txt created previously, store the information in an array using dynamic allocation, write the array to a binary file grade.dat
- The program runs in an interactive mode

  1. TextToDat

  2. Display Dat file

  3. Search and Update.

  4 Quit.

Random access to files

- Functions: fseek() and ftell()
- fseek(): move the file pointer to a position

    fseek(FILE *stream, long offset, int whence);

        stream – file pointer

        offset : length (in bytes).

        whence: constant specifying the position and direction

    - SEEK_SET: from the beginning, and move to the end of the file
    - SEEK_CUR: from the current position of the file pointer and move to the end of the file
    - SEEK_END: move from the end of the file to the beginning

Random access to files

- Function ftell: returns the current position of the file pointer

  long ftell(FILE *stream);

- Function rewind(): set the file pointer to the beginning of the file

  void rewind(FILE *stream);

Exercise

- Write a program that read a part of the file phonebook.dat, for example, from the record a to the record b. Change some information of the records read, and store these records to the file at the position extracted.
- Use dynamic allocation

Solution

```
#include <stdio.h>
#include <stdlib.h>
enum {SUCCESS, FAIL, MAX_ELEMENT = 20};
// the phone book structure
typedef struct phoneaddress {
      char name[20];
      char tel[11];
      char email[25];
}phoneaddress;
int main(void){
   FILE *fp;
   phoneaddress *phonearr;

   int i,n, irc; // return code
   int reval = SUCCESS;
   printf("Read from 2sd data to 3rd data \n");
```

Solution

```
 if ((fp = fopen("phonebook.dat","r+b")) == NULL){
     printf("Can not open %s.\n", "phonebook.dat"); reval = FAIL;
 }
// Memory allocation
phonearr = (phoneaddress *)malloc(2 * sizeof(phoneaddress));
if (phonearr == NULL){
    printf("Memory allocation failed!\n");
    return FAIL;
}
if (fseek(fp,1*sizeof(phoneaddress),SEEK_SET) != 0){
    printf("Fseek failed!\n");
    return FAIL;
}
irc = fread(phonearr, sizeof(phoneaddress), 2, fp);
```

Solution

```
      for (i=0; i<2; i++){
      printf("%s-",phonearr[i].name);
      printf("%s-",phonearr[i].tel);
      printf("%s\n",phonearr[i].email);
    }
     // Modifying some data
    strcpy(phonearr[1].name,"Lan Hoa");
    strcpy(phonearr[1].tel,"0923456");
    strcpy(phonearr[1].email,"lovelybuffalo@hut.edu.vn");
    fseek(fp,1*sizeof(phoneaddress),SEEK_SET);
    irc = fwrite(phonearr, sizeof(phoneaddress), 2, fp);
    printf(" fwrite return code = %d\n", irc);
    fclose(fp); free(phonearr);
    return reval;
}
```

Homework

- Write a program that read Vietnamese-English dictionary from

  http://www.denisowski.org/Vietnamese/vnedict.txt
- Write the data into a binary file using fwrite

Homework: Split and Merge Files

- Use file phonebook.dat (created previously) contains at least 20 records. Write a programs:
- Program **filesplit** has parameters: source file name (.dat); a positive integer N; and the names of two resulting files. **filesplit** will split the source file into 2 files, in which the first file contains N N first records and the second file contains remaining records. Example

  filesplit phone.dat 10 phone1.dat phone2.dat
- Program **filemerge** merges two files into one file

  filemerge phone1.dat phone2.dat phone.dat
- Program **fileread** that reads and displays the records of any .dat file. It will be used to check the correctness of **filesplit** and **filemerge**.

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict