HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

C Programming Basic
Basis

ONE LOVE. ONE FUTURE.

content

- Introduction to the course
- Review of C
- Arrays
- Strings
- Pointers
- Command-line arguments

Introduction to the course

ONE LOVE. ONE FUTURE.

Introduction to the course

- Practical programming in data structures and algorithms topics
- Level: foundation
- Implementing data structures and algorithms
- Applying them to solve real-world problems.
- Programming language: C

Introduction to the course

- Recommended environment:
- Operating system: UNIX
- Compiler: gcc
- Source code editor: Emacs, K-Developer.

Introduction to the course

- Process score: 30%
  - Weekly homework assignments, combined with the quality of completing in-class exercises.
  - Per week: 4-5 students.
- End-of-term: 70%
  - Computer programming exam.
  - Evaluation based on program execution results, not on source code.

HUST

hust.edu.vn fb.com/dhbkhn

Review of C

Review of C

- Parameter:
  -Wall : turn on all alerts
  -c: make object file
  -o: name of output file
  -g: debug information
  -l: library

  > gcc –Wall hello.c –o runhello
  > ./runhello

Review of C

- Topic:
  - Array, String, Pointer Review
  - Character based File operations in UNIX
  - Programming Exercises

Array

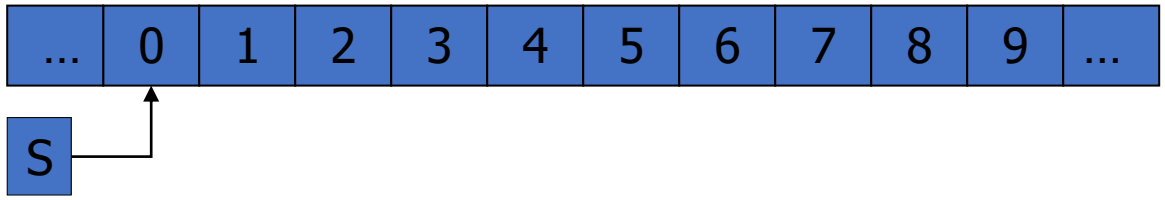
Array

- A block of many variables of the same type
- Array can be declared for any type
  - E.g. int A[10] is an array of 10 integers.
- Examples:
  - list of students' marks
  - series of numbers entered by user
  - vectors
  - matrices

Arrays in Memory

- Sequence of variables of specified type
- The array variable itself holds the address in memory of beginning of sequence
- Example:

double S[10];



- The k-th element of array A is specified by A[k-1] **(0-based)**

Example - reverse

- Example 1.
- Write a program that takes input of an array of 10 integer numbers from the keyboard and prints the entered values in reverse order.

```c
#include <stdio.h>

int main(void)
{
   int i, A[10];

   printf("please enter 10 numbers:\n");
   for (i = 0; i < 10; i++)
      scanf("%d", &A[i]);

   printf("numbers in reversed order:\n");
   for (i = 9; i >= 0; i--)
      printf("%d\n", A[i]);

   return 0;
}
```

Example

- Example 2. Write a program that inputs a string containing only lowercase letters.
  Print the frequency of each alphabetical character in the string, ignoring non-alphabetic characters.

The output for the input line: "hello, world!"

```
The letter 'd' appears 1 time(s).
The letter 'e' appears 1 time(s).
The letter 'h' appears 1 time(s).
The letter 'l' appears 3 time(s).
The letter 'o' appears 2 time(s).
The letter 'r' appears 1 time(s).
The letter 'w' appears 1 time(s).
```

Suggestion:

- Use an array to count the frequency of appearance for each alphabetic character.
- There are 26 lowercase alphabetical characters, create an integer array int count[26].
- The character 'a' corresponds to the first element of the array with index 0.
- Read the string by reading each character as an array or use available functions in the string library.

Example

```c
// maximum size 26
#define ALPHABET_LEN 26

int main(void)
{
   int i = 0, count[ALPHABET_LEN] = { 0 }; // initialize all its elements to zero
   char c = '\0';

   printf("Please enter a line of text: \n");

   /* Read in letter by letter and update the count array */
   c = getchar();
   while (c != '\n' && c >= 0) {
      if (c <= 'z' && c >= 'a')
         ++count[c - 'a']; if (c <= 'Z' && c >= 'A')
         ++count[c - 'A';
      c = getchar();
   }
   for (i = 0; i < ALPHABET_LEN; ++i) {
      if (count[i] > 0)
         printf("The letter '%c' appears %d time(s).\n", 'a' + i, count[i]);
   }
   return 0;
}
```

Example

```c
#include <stdio.h>
#include <string.h>
#define ALPHABET_LEN 26

int main() {
   int i = 0, count[ALPHABET_LEN] = { 0 }; // initialize all its elements to zero
   char s[20], c = '\0';
   printf("Please enter a line of text: \n");
   gets(s);
   for (i = 0; i < strlen(s); i++) {
      c = s[i];
      if (c <= 'z' && c >= 'a') {
         ++count[c - 'a'];
      }
   }
   for (i = 0; i < ALPHABET_LEN; ++i) {
      if (count[i] > 0) {
         printf("The letter '%c' appears %d time(s).\n", 'a' + i, count[i]);
      }

   }
   return 0;
}
```

Exercise

- Exercise 1. Write a program that has
- A function to compare two arrays (having the same number of elements) to see if they match (each element is at the same position in both arrays). If the two arrays match, it returns 1; otherwise, it returns 0.
- A function to check two arrays (with the same elements) for similarity (expansive). Both arrays contain the same elements, but these elements do not need to be in the same positions as in the function above. For example, Array A={2,4,4,5} would match with Array B={4,5,4,2} but not with Array C={5,5,4,2}.
- A function to check two arrays (with the same elements) if the consecutive elements have the same order or not. For example, if Array A[i]<=A[i+1], then Array B must follow the same order as B[i]<=B[i+1]. For instance, Array A={1,3,2,7} would have the same order as Array B={2,7,3,4}.

Exercise

```c
#include <stdio.h>

#define SIZE 5
/*function check if two arays are identical*/
int compare_arrays(int arr1[], int arr2[], int size)
{
   int i = 0;

   for (i = 0; i < size; ++i) {
      if (arr1[i] != arr2[i])
         return 0;
   }

   return 1;
}
```

```c
int main(void)
{
   int input1[SIZE], input2[SIZE], i;

   printf("Please enter a list of %d integers:\n", SIZE);
   for (i = 0; i < SIZE; ++i)scanf("%d", &input1[i]);

   printf("Please enter another list of %d integers:\n", SIZE);
   for (i = 0; i < SIZE; ++i) scanf("%d", &input2[i]);

   if (compare_arrays(input1, input2, SIZE) == 1)
      printf("Both lists are identical!\n");
   else
      printf("The lists are not identical...\n");

   return 0;
}
```

Lab test

- **Lab 01**. Find all perfect square in a sequence
  - Given a sequence of n integers a1, a2, . . ., an. Compute the number Q of perfect squares (the number of type a^2) of that sequence.
- Input
  - Line 1: contains a positive integer n (1 <= n <= 100000)
  - Line 2: contains n positive integer a1, a2, . . ., an (1 <= ai <= 1000000)
- Output
  - Write the value Q

Input	Output
5 3 2 4 7 9	2

Lab test

- Lab 02. Sum Array
  - Given a sequence of integers a1, a2, ..., an. Compute the sum Q of elements of this sequence.
- Input
  - Line 1: contains n (1 <= n <= 10000)
  - Line 2: contains a1, a2, ..., an (-10000 <= ai <= 10000)
- Output
  - Write the value of Q

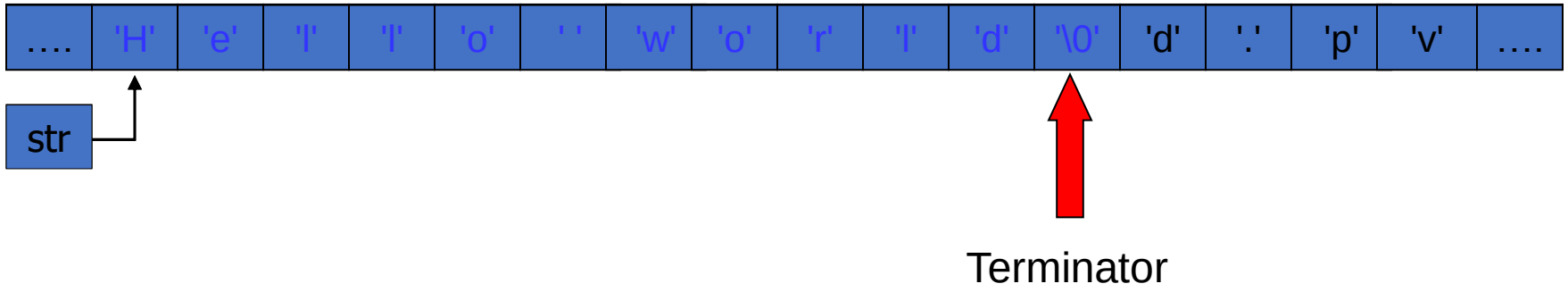
Input	Output
4 3 2 5 4	14

String

Strings

- An array of characters
- Used to store text
- Another way to initialize:

```
char str[] = “Hello world";
```



- In order to hold a string of N characters we need an array of length N + 1
- So the previous initialization is equivalent to

```
char str[] = { 'b', 'l', 'a', 'b', 'l', 'a', '\0' };
char str[] = "blabla";
```

String and character related function

- getchar()
  - c = getchar()
- scanf
  - scanf("%s", str);
- gets()
  - gets(str);

`strlen(const char s[])`
returns the length of s
`strcmp(const char s1[], const char s2[])`
compares s1 with s2
`strcpy(char s1[], const char s2[])`
copies to contents of s2 to s1

Example

- Example 1. write a function that:
  - gets a string and two chars
  - the functions scans the string and replaces every occurrence of the first char with the second one.
- write a program to test the above function
  - the program should read a string from the user (no spaces) and two characters, then call the function with the input, and print the result.
- example
  - input: “papa”, ‘p’, ‘m’
  - output: “mama”

Example

```
/* function that replace all charaters <replace_what> by <replace_with> in string str */
void replace(char str[], char replace_what, char replace_with)
{
   int i;

   for (i = 0; str[i] != '\0'; ++i)
   {
      if (str[i] == replace_what)
         str[i] = replace_with;
   }
}
```

Example

```c
#define STRING_LEN 100

int main(void)
{
   char str[STRING_LEN + 1];
   char replace_what, replace_with, tmp;

   printf("Please enter a string (no spaces)\n");
   scanf("%100s", str);

   printf("Letter to replace: ");
   scanf(" %c", &replace_what);
   do { tmp = getchar(); } while (tmp != '\n');

   printf("Letter to replace with: ");
   scanf(" %c", &replace_with);

   replace(str, replace_what, replace_with);
   printf("The result: %s\n", str);
   return 0;
}
```

Exercise

- Exercise 1. Write a program that reads a string representing a sentence from the user. Then the program displays each word in the sentence on a separate line. A word is a sequence of consecutive characters without containing any whitespace.
- Example:
- Input: "The house nextdoor is very old."
- Result:
- The
- house
- ...

Exercise

- Exercise 2. Write a program that asks the user to enter the number of students in a class, then input the full names of each student in Vietnamese. Display the list of students sorted by their names. For example:
  - Nguyen Bao Anh
  - Tran Quang Binh
  - Vuong Quoc Binh
  - Dao Thi Ha
  - Ngo Anh Vu
- Advanced (optional): Display the maximum count of students with the same name.

Exercise

- Lab 01. Count words
  - Given a Text, write a program to count the number of words (ignore characters SPACE, TAB, LineBreak) of this Text
- Input
  - The Text
- Output
  - Write the number of words

Input	Output
Hanoi University Of Science and Technology School of Information and Communication Technology	12

Exercise

- Lab 02. Text Replacement
  - The text T and two patterns P1, P2 are both strings (without line breaks, with a length not exceeding 1000 characters). Please replace all occurrences of string P1 in T with string P2.
- Input Data
  - Line 1: string P1
  - Line 2: string P2
  - Line 3: text T
- Output:
  - Write the text T after replacement
- Example

Input	Output
AI Artificial Intelligence Recently, AI is a key technology. AI enable efficient operations in many fields.	Recently, Artificial Intelligence is a key technology. Artificial Intelligence enable efficient operations in many fields.

HUST

hust.edu.vn fb.com/dhbkhn

pointer

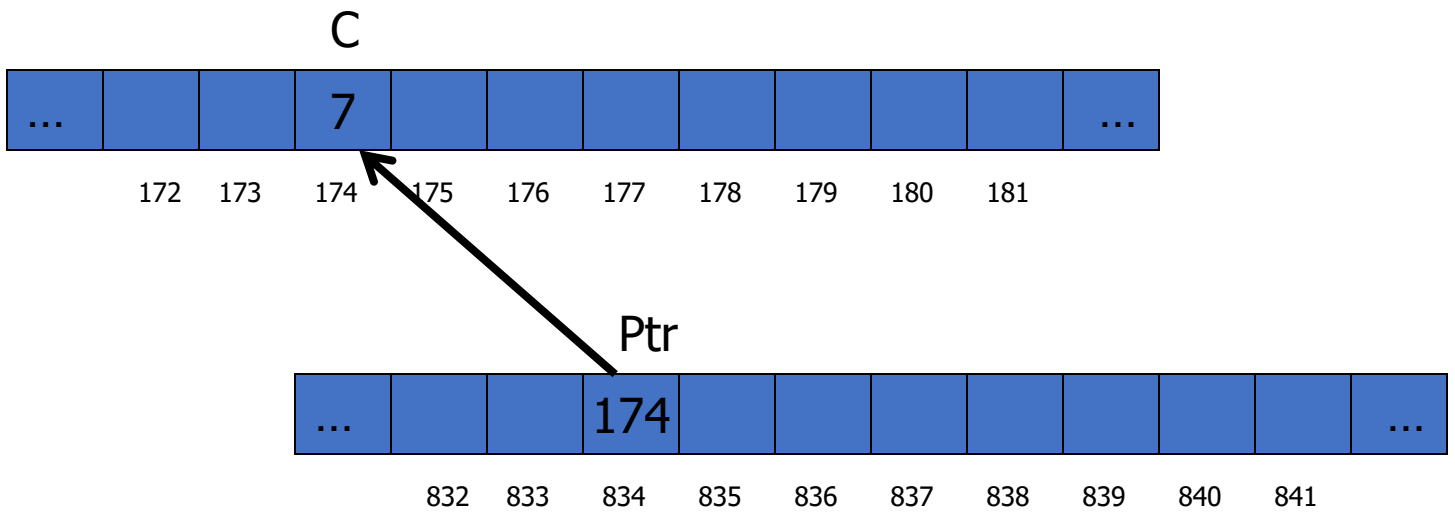
Pointer - Declaration

```
type *variable_name;
```

- A pointer is declared by adding a * before the variable name.
- Pointer is a variable that contains an address in memory.
- The address should be the address of a variable or an array that we defined.

Pointers

- Here ptr is said to *point* to the address of variable c



Referencing and Dereferencing

```
int n;
int* iptr; /* khai báo P là một con trỏ kiểu int */
n = 7;
iptr = &n;

printf(" % d", *iptr); /* Hiển thị ‘7’*/
*iptr = 177;
printf("% d", n); /* Hiển thị ‘177’ */
iptr = 177; /* Phép gán không đúng!! */
```

Example

Example 1. Write a function that accepts a double parameter and returns its integer and fraction parts.

Write a program that accepts a number from the user and prints out its integer and fraction parts, using this function.

```
void split(double num, int* int_part, double* frac_part)
{
   *int_part = (int)num;
   *frac_part = num - *int_part;
}
```

Example

```
int main(void)
{
   double num, fraction;
   int integer;

   printf("Please enter a real number: ");
   scanf("%f", &num);

   split(num, &integer, &fraction);
   printf("The integer part is %d\n", integer);
   printf("The remaining fraction is %f\n", fraction);

   return 0;
}
```

Example

- Example 2. Write a function with the prototype:

```
void replace_char(char *str, char c1, char c2);
```

- It replaces each appearance of `c1` by `c2` in the string `str`.
  Do not use the `[]` operator!
- Demonstrate your function with a program that uses it

```
void replace_char(char* str, char c1, char c2)
{
   if (str == NULL)
      return;

   while (*str != '\0') {
      if (*str == c1) {
         *str = c2;
      }
      ++str;
   }
}
```

Exercise

- Exercise 1. Write a program capable of generating random sentences using the selection technique based on random numbers.
- The program uses four arrays of strings to store articles, nouns, verbs, and prepositions.
- Sentences are created by randomly selecting elements from these arrays and concatenating them in the following order: article, noun, verb, article, and noun.
- The generated sentence needs to start with an uppercase letter and end with a period. The program needs to generate a minimum of 10 sentences.
- Example of array elements:
  - Articles: "the", "a", "one", "some", and "any";
  - Nouns: "boy", "girl", "dog", "town", and "car";
  - Verbs: "drove", "jumped", "ran", "walked", and "skipped";
  - Prepositions: "to", "from", "over", "under", and "on".

HUST

hust.edu.vn fb.com/dhbkhn

Command-line arguments

Command line arguments

- Command line arguments are arguments for the **`main`** function
  - Recall that main is basically a function
  - It can receive arguments like other functions
  - The ‘calling function’ in this case is the operating system, or another program

‘main’ prototype

```
int main(int argc, char* argv[])
```

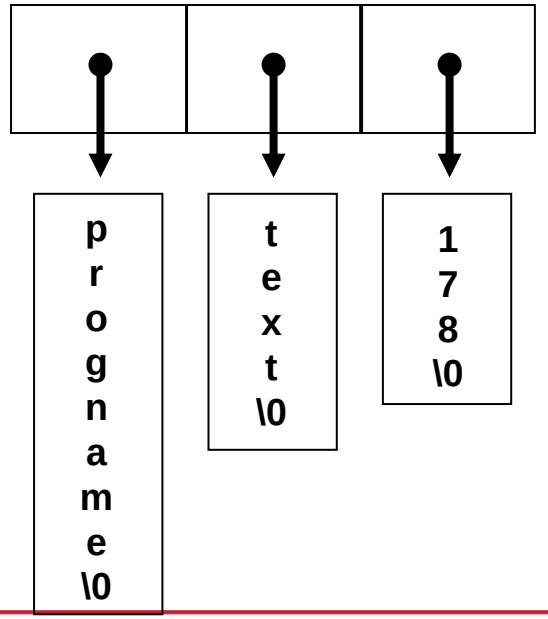
- When we want **`main`** to accept command line arguments, we must define it like this
  - **`argc`** holds the number of arguments that were entered by the caller
  - **`argv`** is an array of pointers to char – an array of strings – holding the text values of the arguments
- The first argument is always the program’s name

‘main’ prototype

```
int main(int argc, char* argv[])
```

```
argc : 3
```

```
argv :
```



Example

- Example 1. Write a program that accepts two numbers as command line arguments, representing a rectangle's height and width (as floating-point numbers).
- The program should display the rectangle's area and perimeter

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[])
{
   double width, height;
   if (argc != 3){
      printf("Wrong number of arguments!\n");
      printf("CORRECT SYNTAX : RECT <WIDTH> <HEIGHT>\n");
      return 1;
   }
   width = atof(argv[1]);
   height = atof(argv[2]);
   printf("The rectangle's area is %f\n", width* height);
   printf("The rectangle's perimeter is %f\n", 2 * (width + height));
   return 0;
}
```

Exercise

- Exercise 1. Reverse the sentence.
- Write a program that allows users to input a sentence as command-line arguments (each word in the sentence is an argument). The program displays the reversed content of the input sentence.
- Example: ./inverse I love HUST
- Output: HUST love I

Exercise

- Exercise 2. Write a program named "sde" that accepts command-line arguments as the coefficients of a quadratic equation ax^2 + bx + c = 0 and solves the equation, then displays the roots on the screen.
- Syntax using sde a b c
- For example: ./sde 1 2 1 yields the result: x1 = x2 = -1