РОЗДІЛ 2.
Рядки

З погляду математики рядок – це послідовність символів. У мові C++ є кілька способів зображення рядків. Далі розглянемо два з них: С-рядки та тип `string`.

2.1. С-рядки

Найпростіша структура даних для зберігання послідовності символів – масив символів. У мові С, від якої походить мова C++, рядок зображується масивом символів, який містить послідовні символи й маркер кінця рядка – символ `'\0'` (нуль-символ).

> **С-рядок** – це зображення послідовності символів у масиві символів з доданим маркером кінця `'\0'`.

Значенням С-рядка (англ. – C-string value) є послідовність символів від його початку, обмежена найближчим `'\0'`. Під час виконання програми С-рядок, будучи масивом символів, ідентифікується *адресним виразом*, що задає адресу даних типу `char`. Послідовність символів, розташовану, починаючи з адреси, заданої адресним виразом, називатимемо **рядковим значенням**, або просто **рядком**.

У мові C++ для роботи із С-рядками є спеціальні засоби, реалізовані як у самій мові, так і бібліотечні. Проте сучасні реалізації мови C++ надають інший спосіб зображення рядків, набагато зручніший і безпечніший, ніж С-рядки (див. підрозд. 2.2). Тому розглянемо лише деякі мовні особливості, зв'язані із С-рядками, і кілька бібліотечних функцій.

Ініціалізація рядковою константою

Рядкова константа, наприклад `"abc"`, зображується в пам'яті як С-рядок, тобто масивом із чотирьох (а не трьох!) символів. До байтів із символами `'a'`, `'b'`, `'c'` *додається ще один* – з нуль-символом `'\0'`. У пам'яті програми константа `"abc"` задається *адресою першого байта* цієї послідовності байтів. Звідси рядковою константою можна ініціалізувати масив символів або присвоїти чи ініціалізувати нею вказівник типу `char*`.

Приклад. Ініціалізуємо вказівник на символи й масив символів рядковими константами.

```
char *p = "123"; char s[6] = "abc";
```

Вказівник `p` встановлено на перший із чотирьох послідовних байтів, що містять символи `'1'`, `'2'`, `'3'`, `'\0'`. Елементи масиву `s` від `s[0]` до `s[2]` мають значення `'a'`, `'b'`, `'c'`, а від `s[3]` до `s[5]` – `'\0'`. ◂

Ініціалізація масиву символів рядковою константою та присвоювання цієї константи вказівнику типу `char*` мають, як мінімум, одну відмінність. Обидві рядкові константи `"123"` та `"abc"` з прикладу задають *незмінювані послідовності символів* типу `char const[4]`. Проте інструкція

```
char s[6] = "abc";
```

створює *змінну* `s` (масив із 6 байтів) та ініціалізує його значеннями `'1'`, `'2'`, `'3'`, `'\0'`. Елементи масиву є *символьними змінними, які можна змінювати*. Наприклад, значення масиву `s` можна зробити порожнім рядком, присвоївши `s[0]='\0'`. Водночас інструкція

```
char *p = "123";
```

створює змінну-вказівник і встановлює його на перший байт С-рядка `"123"`. Вказівник `p` насправді вказує на послідовність *незмінюваних* символів. Спроба під час виконання програми змінити символ у межах цієї послідовності буде проігнорована або призведе до аварійного закінчення – залежно від версії та настройок компілятора.

Якщо розмір масиву в ініціалізації рядковою константою не вказано, то з додатковим `'\0'` він буде на 1 більше ніж довжина константи. Наприклад, після оголошення

```
char a[]="abc";
```

маємо `sizeof(a)`=4. Водночас ініціалізація

```
char a[]={'a','b','c'};
```

означає масив із трьох елементів, в якому немає `'\0'` (і це є *потенційно небезпечним*).

- ✓ Якщо масив символів має розмір *N*, то довжина ініціалізуючого С-рядка повинна бути не більше *N*–1, інакше компілятор повідомить про помилку.
- ✓ Якщо всі елементи масиву символів мають значення, відмінні від `'\0'`, то значення С-рядка, зображеного в масиві, закінчується *невідомо де*, і це може мати *непередбачувані наслідки*.

Операція `cout<<s`, де `s` – ім'я масиву, тип якого не є символьним, виводить значення вказівника `s`, тобто деяку адресу. Проте, якщо `s` – масив символів, то операція виводить його рядкове значення. Узагалі, якщо *AE* – адресний вираз типу `char*`, то операція вигляду `cout<<`*AE* виводить рядкове значення С-рядка – від байта, на який указує адресний вираз, до байта перед найближчим символом `'\0'`. Наприклад, якщо вказівник `p` встановлено на рядкову константу `"12345"`, то вираз

```
cout << p << '::' << p+2
```

виводить символи `12345::345`.

Під час уведення в масив символів, тобто в *змінну*, наприклад `s`, операція вставлення з потоку `cin>>s`, як і для інших типів, пропускає порожні символи (пропуск, табуляція, кінець рядка), переписує в масив `s` непорожні до найближчого порожнього й дописує до них символ `'\0'`.

✓ Кількість уведених символів *не контролюється*, тому, якщо заповнюються байти за межами масиву `s`, можливі *непередбачувані наслідки*.

Приклад. Нехай діє оголошення `char s1[10];`. Під час виконання `cin>>s1` у вхідному потоці пропускаються порожні символи, потім непорожні записуються в масив до появи порожнього. Якщо цих символів більше дев'яти, то символ-завершувач дописується вже за межами масиву `s1`, і це може бути небезпечним. ◂

✓ Оголошуючи масив символів, необхідно забезпечити, щоб його розмір був достатнім для обробки можливих вхідних даних.

Узагалі, якщо *AE* – адресний вираз типу `char*`, що вказує на *змінювані дані*, то операція вигляду `cin>>`*AE* записує непорожні символи в послідовні байти, починаючи з байта, на який указує вираз *AE*. *І це теж може бути небезпечним.*

Спроба ввести символи в незмінюваний рядок, на який установлено вказівник, призводить до аварійного закінчення, адже *не можна змінити незмінювані дані*. Наприклад, таким чином:

```
char *s="123456789";
cin>>s; // помилка під час виконання
```

Параметри головної функції

Запустити програму на виконання можна не тільки засобами системи програмування або файлового менеджера. Будь-яка операційна система дозволяє запустити програму в **командному рядку**, або **рядку виклику**.

У рядку виклику вказується шлях до файлу з програмою та, можливо, ще кілька слів (послідовностей непорожніх символів). Операційна система зчитує цей рядок, визначає кількість *n* слів у ньому й створює масив з *n*+1 вказівників на символи. Вона також послідовно записує слова з рядка виклику з обмежувачами `'\0'` у масив символів і встановлює вказівники на початки цих слів. Вказівник з індексом *n* отримує нульове значення, позначене в мові C++ константою `NULL`. Слова з рядку виклику програми називають **аргументами виклику**.

У заголовку функції `main()` можна записати два параметри: перший типу `int`, другий типу `char**` або `char*[]` (тобто *вказівник на вказівники на символи*). Тоді на початку виконання головної функції значеннями параметрів стають, відповідно, кількість слів у рядку виклику й адреса першого елемента в масиві вказівників на рядки. За традицією, параметри програми мають імена `argc` (типу `int`) і `argv` (типу `char**`), як на рис. 2.1.

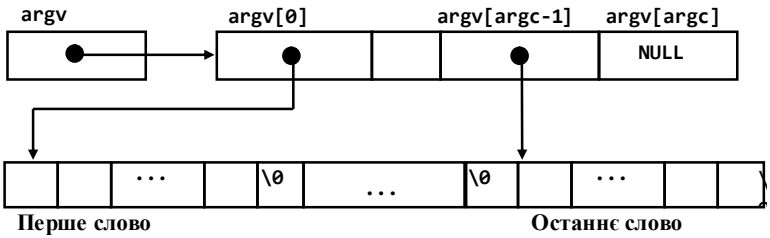

Рис. 2.1. Зберігання параметрів

Приклад. Програма виводить на екран слова, отримані з командного рядка. Слова є значеннями С-рядків, які позначаються виразами `argv[0]`, `argv[1]`, …, `argv[argc-1]`.

```
#include <iostream>
usung std::cout;
int main(int argc, char **argv)
{ for (int i=0; i<argc; ++i)
    cout << argv[i] << '\n';
  return 0;
}
```

Якщо запустити цю програму із системи програмування або файлового менеджера, то виходом буде лише ім'я виконуваного файлу (зі шляхом до нього у файловій системі). ◂

Вправи

2.1. Що буде виведено за програмою?

```
#include <iostream>
using namespace std;
void printSuffs(char * p){
  while(*p) cout << p++ << ' ';
  cout << endl;
}
int main(){
  printSuffs("gambit");
  system("pause");
  return 0;
}
```

2.2. Написати програму, яка перевіряє кількість аргументів виклику й друкує повідомлення `Too many`, якщо аргументів більше ніж 3; повідомлення `Not enough`, якщо їх менше ніж 3; якщо ж їх 3, то повідомлення `OK` і самі аргументи.

2.2. Тип string

У мові C++ для зображення рядків є стандартний бібліотечний тип `string`. Для роботи з ним необхідно включити файл `<string>` і скористатися простором імен `std`. Тип `string` насправді є *класом*, а змінні цього типу – *об'єктами* (див. розд. 3).

Аналогічно С-рядкам, рядкове значення змінної типу `string` – це зображена нею послідовність символів.

Змінні типу `string` можна ініціалізувати за допомогою рядкових констант, С-рядків, ідентифікованих іменем масиву символів або вказівником на символи, та інших змінних типу `string`. Наприклад, інструкції

```
char cs[100] = "123";
string s0, s1("ABC"), s2(s1), s3 = cs;
```

надають неініціалізованій змінній `s0` як значення порожній рядок, змінним `s1` і `s2` – рядок `ABC`, змінній `s3` – рядок `123`.

✓ Довжина рядкового значення змінних типу `string` визначається не за символом `'\0'`, а зберігається окремо в пам'яті,

тому, на відміну від С-рядків, символ '\0' може входити в рядкове значення змінної типу string.

2.3. Операції з рядками

Розглянемо кілька типових операцій, специфічних для рядків.

Визначення довжини рядка, тобто кількості символів рядка. Наприклад, довжина рядка ABC дорівнює 3, порожнього рядка – 0.

Конкатенація рядків. Конкатенація двох рядків – це результат дописування другого рядка в кінці першого. Наприклад, конкатенацією рядків ABC і 12 є ABC12.

Лексикографічне порівняння рядків. Це порівняння рядків, засноване на порядку символів у ASCII-таблиці й аналогічне звичайним словникам. Наприклад, рядок bca менше ніж cw або A, але більше ніж abcd та bc.

До рядків застосовні операції пошуку входження символу чи підрядка в рядок, а також копіювання рядкового значення, уведення з потоку >>, виведення в потік <<.

Зазначені операції реалізовані в мові C++ як для С-рядків, так і для рядків типу string. Далі розглянемо лише деякі бібліотечні засоби обробки рядків. Детальнішу інформацію шукайте в довідкових системах середовищ програмування та інших джерелах.

Обробка С-рядків

Функції обробки С-рядків оголошено в бібліотечному файлі <string.h> (включати потрібно файл <cstring>). Розглянемо лише деякі з них.

Функція із заголовком

```
int strlen(char * s)
```

повертає ціле число – довжину рядкового значення s.

Функція із заголовком

```
char* strcat(char* s1, const char* s2)
```

реалізує дописування рядків: рядкове значення s2 дописується в кінець С-рядка s1, адресою початку результуючого С-рядка є s1, який і повертається функцією. Ця функція небезпечна, оскільки не контролює, чи достатньо ділянки пам'яті, адресованої s1, щоб умістити результат конкатенації. Тому в деяких версіях мови C++ замість неї рекомендовано використовувати безпечніший аналог strcat_s.

Функція із заголовком

```
int strcmp(const char* s1, const char* s2)
```

порівнює рядкові значення лексикографічно й повертає ціле значення – від'ємне, якщо `s1` менше, ніж `s2`; 0, якщо вони рівні, і додатне, якщо `s1` більше, ніж `s2`. Функція `strncmp(s1,s2,n)` робить те саме, тільки порівнює не більше ніж `n` перших символів рядків. Крім звичайного лексикографічного порівняння, у бібліотеці наявні функції, що забезпечують порівняння без урахування регістра символів та/або з урахуванням регіональної (локальної) специфіки.

Функція із заголовком

```
char* strcpy(char* s1, const char* s2)
```

записує вміст С-рядка `s2` на місце С-рядка `s1` і повертає адресу початку останнього, тобто `s1`. Аналогічно функції `strcat`, ця функція теж потенційно небезпечна. Її безпечніший аналог – функція `strcpy_s`. Зазначимо, що звичайне присвоювання С-рядків, заданих вказівниками, не копіює їх вміст і може мати непередбачувані наслідки (наприклад, помилки в обробці динамічних даних – див. розд. 4).

Операції введення з потоку `>>` і виведення в потік `<<` для С-рядків було розглянуто в підрозд. 2.1.

Обробка даних типу string

Довжину рядкового значення змінної типу `string` повертає функція `length()`. Наприклад, після означення рядків

```
string s0, s1("ABC");
```

вираз `s0.length()` має значення 0; `s1.length()` – значення 3.

- ✓ Якщо тип є класом, то виклик функції, означеної для типу, який застосовується до змінної-об'єкта, записується після імені змінної через крапку.

Для типу `string` означено операції присвоювання `=`, конкатенації `+`, дописування `+=`, порівнянь `==`, `!=`, `<`, `<=`, `>`, `>=`, уведення з потоку `>>`, виведення в потік `<<`, індексації `[]` та інші. Розглянемо деякі з них.

Присвоювання реалізує операцію копіювання рядків. Змінній типу `string` можна присвоїти або іншу змінну цього типу, або С-рядок, або символ. Наприклад, для змінних, означених вище, можливі вирази присвоювання `s0=s1`, `s0=cs`, `s0="QWE"`, `s0='A'`. Перші три присвоювання копіюють у змінну `s0` рядкові значення

виразів праворуч, і вони стають її рядковими значеннями, останнє робить рядковим значенням змінної `s0` рядок з одного символу.

Операція конкатенації `+` застосовна до змінних типу `string`, С-рядків і символьних значень; хоча б один із двох її операндів має бути об'єктом типу `string`. Наприклад, після ініціалізації змінних

```
string s1="ABC", s2="12"; char cs[100]="UVWX";
```

вираз `s1+s2` має рядкове значення `ABC12`, вираз `s1+cs` – `ABCUVWX`, `cs+s1` – `UVWXABC`, `cs[0]+s2+"MN"` – `U12MN`. Аналогічним є **дописування**: після `s1+=s2` об'єкт `s1` має рядкове значення `ABC12`, а після `s2+=cs[0]` об'єкт `s2` має рядкове значення `12U`.

Операції порівняння застосовуються до об'єктів типу `string` або С-рядків; хоча б один з операндів має бути об'єктом типу `string`. Результатом є лексикографічне порівняння відповідних рядкових значень. Наприклад, після ініціалізації

```
string s1="AB", s2="ABC"; char cs[100]="a";
```

вирази `s1<s2` і `s2<=cs` мають значення "істина", а вираз `s1+s2>cs` – "хибність".

Операції введення та виведення. Операція введення з потоку `>>` у змінну типу `string` пропускає в потоці порожні символи та створює рядкове значення з послідовності непорожніх символів до найближчого порожнього або до кінця потоку. Кількість символів, що потрапляють у рядок, *практично необмежена*.

Функція `getline` (аргументами в її виклику є потік і об'єкт-рядок) уводить із потоку в рядок усі символи до найближчого символу кінця рядка. Доступним у потоці стає символ, наступний після цього кінця рядка.

Операція `<<` виводить у потік рядкове значення виразу типу `string`.

Приклад. Якщо до кінця рядка потік `cin` містить символи `1 2 3`, то за інструкціями

```
cin >> s1; getline(cin,s2);
```

об'єкт-рядок `s1` отримує рядкове значення `1`, `s2` – `□2□3` (тут символ □ позначає пропуск). Після цього інструкція

```
cout << s1+s2+' '+s1;
```

виводить у потік `cout` послідовність символів `1□2□3□1`. ◂

Вираз вигляду `s[i]` з **операцією індексації** позначає символьну змінну (елемент об'єкта-рядка `s`) або її значення. Наприклад, після присвоювання `s2="012"` вираз `s2[0]` має значення `'0'`, а присвоювання `s2[2]='X'` надає `s2` значення `01X`.

✓ Програміст має стежити, щоб у виразі вигляду `s[i]` значення `i` було в межах від `0` до `s.length()-1`, інакше можливі непередбачувані наслідки.

Порівняно із С-рядками клас `string` має суттєві переваги:

– це *повноцінний тип*, який дозволяє записувати вирази типу `string` (зокрема повертати їх із функцій);

– оператори мови C++ (`=`, `+`, `>>` тощо), означені для цього класу, забезпечують *зручний запис операцій* з рядками;

– операції з рядками, означені для класу `string`, *убезпечують від "залізання" в інші змінні*.

Тип `string` має й певні недоліки, але їх розгляд виходить за межі цього посібника.

Вправи

2.3. Написати власні варіанти функцій: а) `strlen`; б) `strcmp` і `strncmp`; в) `strchr` і `strrchr`; г) `strcpy` і `strncpy`; д) `strcat` і `strncat`; е) `strstr`; ж) `strpbrk`; з) `strspn` і `strcspn`; и) `strcpy_s`.

2.4. Написати програму, яка виводить розмір змінних типу `string`, ініціалізованих рядковими константами `""`, `"0"`, `"01234567"`, `"0123456789012345678 90"`.

2.5. Написати функцію, що відсікає всі пропуски в кінці рядка[3].

2.6. Написати функцію, що дзеркально перевертає рядок.

2.7. Написати функцію визначення, чи є рядок *паліндромом*, тобто симетричною послідовністю символів.

2.8. Написати функцію визначення, чи є рядок подвоєним рядком, тобто має вигляд *ww*, де *w* – деяка послідовність символів, можливо, порожня.

2.9. Рядок містить слова, відокремлені одне від одного проміжками (у довільній кількості). Написати функцію, яка:
а) визначає кількість слів у рядку;

[3] У цій і наступних задачах можна скористатися як С-рядками, так і об'єктами типу `string`.

б) визначає довжину найдовшого слова рядка;
в) дзеркально перевертає кожне слово рядка;
г) вилучає з рядка всі слова, що містять менше п'яти літер.

2.10. Написати функцію, яка визначає множину символів, наявних у рядку.

2.11. Написати функцію, яка визначає множину символів, що входять у рядок рівно по одному разу.

2.12. Написати функцію, яка за двома рядками визначає:
а) чи збігаються множини символів, наявних у заданих рядках;
б) чи є множина символів першого рядка підмножиною множини символів другого.

Контрольні запитання

2.1. У чому полягає особливість зберігання послідовності символів, заданих рядковою константою?

2.2. Що таке С-рядок?

2.3. У чому полягає відмінність ініціалізації масиву символів рядковою константою й послідовністю символьних констант у фігурних дужках?

2.4. У чому полягає відмінність ініціалізації масиву символів рядковою константою та присвоювання цієї константи вказівнику типу `char*`?

2.5. Чи може головна функція мати параметри? Якщо так, то які?

2.6. Опишіть, як параметри функції `main` отримують значення.

2.7. Які оператори мови C++ застосовні до операндів типу `string`?

2.8. Чи можливе присвоювання С-рядків? Чи можливе присвоювання об'єктів типу `string`?

2.9. Чи може рядкове значення С-рядка містити символ `'\0'`?

2.10. Чи може рядкове значення об'єкта типу `string` містити символ `'\0'`?