**Практична робота №9.** Написання програм з використанням шаблонів функцій

**Завдання.**

Написати на мові С++ програму введення даних для розрахунку формул з передачею параметрів за різними механізмами.

1. Ознайомитися з теоретичною частиною.
2. Написати програму, яка використовує дві шаблонні функції.

2.1. Створити шаблон функції, яка міняє місцями значення двох змінних. Викликати функцію для змінних цілого, дійсного, символьного типів. Вивести результати до і після обміну.

2.2. Створити шаблон функції, яка знаходить максимальне значення. Викликати функцію для змінних цілого, дійсного, символьного типів. Вивести результати до і після обміну.

1. Вхідні дані ввести, а результати вивести, використовуючи потокове введення-виведення даних. В першому рядку кожної програми записати

*// ОПІ-41 Група № Прізвище Номер ЛР*

1. Результати надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді cpp-файлу з іменем у форматі

**<Номер групи><Номер лабораторної><Прізвище англійською>**

Наприклад, МІВТ-МНТ-ЕТ-41-07Ivanov.cpp.

Іншим рішенням є надсилання поштою посилання на текст програми за URL адреси, яку надає C++Shell, вказавши в темі листа, номер групи прізвище студента та номер ПР.

В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№5".

**Строк відсилки ЛР для МІВТ/ МНТ/ЕТ - 41 08.11.2021**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№7 В темі листа вказати, номер групи, прізвище студента, номер ПР та фразу "Запитання".

**Теоретичні відомості.**

***Тип даних char***

Змінна типу char займає 1 байт. Однак, замість конвертації значення типу char в ціле число, воно інтерпретується як ASCII-символ.

**ASCII** (від англ. “**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange”) — це американський стандартний код для обміну інформацією, який визначає спосіб представлення символів англійської мови (+ декілька інших) у вигляді чисел від 0 до 127. Наприклад: код букви 'а' — 97, код букви 'b' — 98. Символи завжди поміщаються в одинарні лапки.

Символи від 0 до 31 в основному використовуються для форматування виводу. Символи від 32 до 127 використовуються для виведення. Це літери, цифри, знаки пунктуації, які більшість комп’ютерів використовує для відображення тексту (англійською мовою).

Наступні два оператори виконують однакові дії - присвоюють змінним типу char ціле число 97:

**char ch1(97);** // ініціалізація змінної типу char цілим числом 97

**char ch2('a');** // ініціалізація змінної типу char символом 'a' (97)

При використанні фактичних чисел для представлення символів (з таблиці ASCII) потрібна уважність. Наступні два оператори виконують різні дії:

**char ch(5);** // ініціалізація змінної типу char цілим числом 5

**char ch('5');** // ініціалізація змінної типу char символом '5' (53)

***Оператор static\_cast***

Якщо ви хочете вивести символи у вигляді цифр, а не у вигляді букв, то потрібно повідомити cout виводити змінні типу char в вигляді цілочисельних значень. Можна присвоїти змінній типу int змінну типу char і вивести її, але це не дуже хороший спосіб:

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**char ch(97);**

**int i(ch); // присвоюємо значенням змінної ch змінній типу int**

**cout << i << endl; // виводимо значення змінної типу int**

**return 0;**

**}**

Результат:

97

Кращим способом є конвертація змінної з одного типу даних в інший за допомогою **оператора static\_cast**.

**Синтаксис static\_cast виглядає наступним чином:**

static\_cast<новий\_тип\_даних>(вираз)

static\_cast приймає значення з (вираз) в якості вхідних даних і конвертує його у вказаний вами <новий\_тип\_даних>.

Приклад використання оператора static\_cast для конвертації типу char в тип int:

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**char ch(97);**

**cout << ch << endl;**

**cout << static\_cast<int>(ch) << endl;**

**cout << ch << endl;**

**return 0;**

**}**

Результат виконання програми:

a  
97  
a

**static\_cast** приймає (вираз) в якості вхідних даних. Якщо ми використовуємо змінну (в виразі), то ця змінна змінює свій тип тільки в інструкції з оператором static\_cast. Процес конвертації ніяк не впливає на вихідну змінну з її значенням! В наведеному вище прикладі, змінна ch залишається змінною типу char з колишнім значенням, чому є підтвердженням останній оператор з cout.

В static\_cast немає ніякої перевірки діапазону, тому при використанні занадто великих або занадто маленьких чисел для конвертованого типу, то відбудеться **переповнення**.

***Введення символів***

Наступна програма просить користувача ввести символ. Потім вона виводить цей символ і його ASCII-код:

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**cout << "Input a keyboard character: ";**

**char ch;**

**cin >> ch;**

**cout << ch << " has ASCII code " << static\_cast<int>(ch) << endl;**

**return 0;**

**}**

Результат виконання програми:

Input a keyboard character: q  
q has ASCII code 113

Зверніть увагу, що навіть якщо cin дозволить вам ввести декілька символів, змінна ch буде зберігати тільки перший символ (саме він і розміщується у змінній). Інша частина користувацького введення залишиться у вхідному буфері, який використовує cin і буде доступна для використання наступним викликам cin.

Приклад:

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**cout << "Input a keyboard character: ";**

**// припустимо, що користувач ввів "abcd"**

**char ch;**

**cin >> ch;**

**// ch = 'a', "bcd" залишається у вхідному буфері**

**cout << ch << " has ASCII code " << static\_cast<int>(ch) << endl;**

**// Звернуть увагу, що наступний cin не просить**

**// користувача що-небудь ввести, дані підтягуються з**

**// вхідного буферу!**

**cin >> ch; // ch = 'b', "cd" залишається в буфері**

**cout << ch << " has ASCII code " << static\_cast<int>(ch) << endl;**

**return 0;**

**}**

Результат виконання програми:

Input a keyboard character: abcd  
a has ASCII code 97  
b has ASCII code 98

***Розмір, діапазон і знак типу сhar***

В С++ для змінних типу char завжди виділяється 1 байт. За замовчуванням, char може бути як **signed, так і unsigned** (хоча зазвичай signed). Якщо ви використовуєте char для зберігання ASCII-символів, то вам не потрібно вказувати знак змінної (так як і signed, і unsigned можуть містити значення від 0 до 127).

Але якщо ви використовуєте тип char для зберігання невеликих цілих чисел, то тоді слід уточнити знак. Змінна типу char signed може зберігати числа від -128 до 127. Змінна типу char unsigned має діапазон від 0 до 255.

***Що використовувати: ‘\n’ чи endl (std::endl)?***

При використанні cout, дані для виведення можуть поміщатися в буфер, тобто cout може не відправляти дані відразу ж на виведення. Замість цього він може залишити їх при собі на деякий час. Це робиться в цілях підвищення продуктивності.

І '\n', і endl обидва переносять курсор на наступний рядок. Тільки endl ще гарантує, що всі дані з буферу будуть виведені, перед тим, як продовжити.

Коли використовувати '\n', а коли endl?

Використовуйте endl, коли потрібно, щоб ваші дані виводилися відразу ж (наприклад, під час запису в файл або при оновленні індикатора стану будь-якого процесу). Зверніть увагу, це може спричинити за собою незначне зниження продуктивності, особливо якщо запис на пристрій відбувається повільно (наприклад, запис файлу на диск).

Використовуйте '\n' у всіх інших випадках.

***Різниця між одинарними і подвійними лапками при використанні з символами***

Символи завжди поміщаються в одинарні лапки (наприклад, 'а', '+' чи '5'). Змінна типу char представляє тільки один символ (наприклад, літеру а, символ + чи число 5). Некоректним записом буде:

**char ch('56'); // змінна типу char може містити тільки 1 символ**

Текст, який знаходиться в подвійних лапках, називається рядком (наприклад, "Hello, world!"). **Рядок** (**тип string**) — це набір послідовних символів.

Ви можете використовувати літерали типу string в коді:

**cout << "Hello, world!";**

**// "Hello, world!" - це літерал типу string**

**Виклик функції з використанням параметра-посилання** здійснює передачу до функції не самої змінної, а тільки посилання на неї. У цьому випадку забезпечується доступ до переданого параметра за допомогою визначення його альтернативного імені, бо посилання є синонімом імені параметра. Тоді всі дії, що відбуваються над посиланням, є діями над самою змінною. Такий спосіб передачі параметрів і повернення результату передбачає запис у списку фактичних параметрів імені змінної, а у списку формальних — параметрів-посилань. Наприклад:

//-------------- **використання параметра-посилання**

**void fun (int &p)** //---------- функция fun()

**{ ++p;**

**cout <<** **"p = " << p << endl;** **}**

**void main** **( )**

**{ int x** **= 10;**

**fun (x);** //---------- вызов функции fun()

**cout <<** **"x="** **<<** **x << endl; }**

Одержимо результат попереднього фрагмента, тобто **р = 11 і х=11.**

При застосуванні вказаних вище параметрів у функцію передаються не значення змінних, а їхні адреси, тому можливо міняти значення цих змінних усередині функції і передавати за її межі (в інші функції). Коли необхідно, щоб деякі параметри не змінювали свої значення всередині функції, їх слід оголосити як параметри-константи, використовуючи модифікатор **const**.

**Покажчики** — це змінні, котрі містять адресу пам’яті, розподіленої для об’єкта відповідного типу. При оголошенні змінної-покажчика слід вказати тип даних, адресу яких буде містити змінна, та ім’я покажчика з символом «\*».

Загальний формат опису покажчика має вигляд:

**тип \*ім’я;**

де **тип** — тип значень, на який вказує покажчик;  
**ім’я** — ім’я змінної-покажчика;  
«\*» — операція над типом, що читається «покажчик на тип».

Наприклад:

**int \*рn** – покажчик на ціле значення;  
**float \*pf1, \*pf2;** — два покажчики на дійсні значення.

Покажчики не прив’язують дані до якого-небудь визначеного імені змінної і можуть містити адреси будь-якого неіменованого значення. Існує адресна константа **NULL**, що означає порожню адресу.

Мова C++ налічує лише дві операції, які стосуються адрес змінних, а саме:

**«&»** — **операція взяття адреси** («адреса значення»);

**«\*»** — **операція розіменування** («значення за адресою»).

Операція взяття адреси **«&»** застосовується разом зі змінною і повертає адресу цієї змінної. Операція розіменування «\*» використовується разом з покажчиками і бере значення, на яке вказує змінна-покажчик, розташована безпосередньо після символу «\*».

Оголошення покажчиків можна здійснити одним з таких способів:

**<тип> \*ptr;**  
**<тип> \*ptr = <змінна-покажчик>;**  
**<тип> \*ptr = &<ім’я змінної>;.**

Наприклад:  
**int \*ptx, b; float у;** — оголошені змінна-покажчик **ptx** та змінні **b і у**;

**float \*sp = &у;** — покажчику **sp** присвоюється адреса змінної **у**;

**float \*р = sp;** — покажчику **р** присвоюється значення (адреса значення), яке міститься в змінній **sp**, тобто адреса змінної **у**.

При оголошенні покажчиків символ «\*» може знаходитися перед ім’ям покажчика або відразу після оголошення типу покажчика і поширювати свою дію тільки на одну змінну-покажчик, перед якою він записаний:

**long \*pt;   long\*Uk;   int \*ki, x, h;** — оголошення описів.

**Виклик функції з використанням параметра-посилання** здійснює передачу до функції не самої змінної, а тільки посилання на неї. У цьому випадку забезпечується доступ до переданого параметра за допомогою визначення його альтернативного імені, бо посилання є синонімом імені параметра. Тоді всі дії, що відбуваються над посиланням, є діями над самою змінною. Такий спосіб передачі параметрів і повернення результату передбачає запис у списку фактичних параметрів імені змінної, а у списку формальних — параметрів-посилань. Наприклад:

//-------------- **використання параметра-посилання**

**void fun (int &p)** //---------- функция fun()

**{ ++p;**

**cout <<** **"p = " << p << endl;** **}**

**void main** **( )**

**{ int x** **= 10;**

**fun (x);** //---------- вызов функции fun()

**cout <<** **"x="** **<<** **x << endl; }**

Одержимо результат попереднього фрагмента, тобто **р = 11 і х=11.**

При застосуванні вказаних вище параметрів у функцію передаються не значення змінних, а їхні адреси, тому можливо міняти значення цих змінних усередині функції і передавати за її межі (в інші функції). Коли необхідно, щоб деякі параметри не змінювали свої значення всередині функції, їх слід оголосити як параметри-константи, використовуючи модифікатор **const**.

В C++ передача за посиланням може здійснюватися двома способами:

* використовуючи безпосередньо посилання;
* за допомогою вказівників.

Синтаксис передачі з використанням посилань має на увазі застосування як аргумент посилання на тип об'єкта. Наприклад, функція

**double** Glue ( **long**& x, **int**& y ) ;

одержує два посилання на змінні типу **long** і **int**. При передачі у функцію параметра за посиланням компілятор автоматично передає у функцію адресу змінної, зазначеної в якості аргументу. Ставити знак амперсанта перед аргументом у виклику функції не потрібно. Наприклад, для попередньої функції виклик з передачею параметрів за посиланням виглядає в такий спосіб:

c = Glue (a, b) ;

Приклад прототипу функції при передачі параметрів через вказівник:

**void** Setnumber ( **int**\*, **long**\* ) ;

Тоді виклик функції має наступний вигляд:

Setnumber (&n,&a ) ;

Приклад функції, яка приймає в якості параметра дві змінні, та міняє їх місцями(параметри передаються з використанням посилань):

void swap(int &x, int &y)

{

int temp=x;

x=y;

y=temp;

}

int \_tmain()

{

int a=3,b=4;

swap(a,b);

cout<<"a="<<a<<"b="<<b<<'\n';

system("pause");

return 0;

}

Приклад функції, яка приймає в якості параметра дві змінні, та міняє їх місцями(параметри передаються з використанням вказівників):

void swap (int \*x, int \*y)

{

int temp=\*x;

\*x=\*y;

\*y=temp;

}

int \_tmain()

{

int a=3,b=4;

swap (&a,&b);

cout<<"a="<<a<<"b="<<b<<'\n';

system("pause");

return 0;

}

Якщо передати параметри за значенням, зміни не будуть збережені, бо в функцію буде передано копії змінних, а не їх адреси.