Практична робота №5
Основні поняття мови програмування С.
Оператори розгалуження програми у мові С

2.1. Мета роботи

Вивчити основні поняття мови програмування С, операції, стандартні функції, оператори розгалуження програм.

2.2. Теоретичні відомості

2.2.1.Мова С[1]

Алфавіт мови С

Програма в мові С записується символами базового алфавіту (*basic character set*), який містить:

- 26 великих літер:
  A B C D E F G H I J K L M
  N O P Q R S T U V W X Y Z
- 26 маленьких літер:
  a b c d e f g h i j k l m
  n o p q r s t u v w x y z
- 10 десяткових цифр:
  0 1 2 3 4 5 6 7 8 9
- 29 спеціальних символів:
  ! " # % & ' ( ) * + , - . / :
  ; < = > ? [\ ] ^ _ { | } ~
- символ пробілу, табулювання та переходу на новий рядок.

Із символів складаються базові елементи мови або лексеми (*tokens*), ними можуть бути:

- ключові слова (*key words*);
- ідентифікатори (*identifiers*);
- константи (*constants*);
- стрічкові літерали (*string literals*);
- знаки пунктуації (*punctuators*).

У свою чергу, лексеми є частиною виразів, а з виразів складаються інструкції та оператори. При компіляції програми на С з програмного коду виділяються лексеми максимальної довжини, що містять допустимі символи. Якщо в програмі є неприпустимий символ, то лексичний аналізатор (або компілятор) видасть помилку, і компіляція програми виявиться неможливою.

---

[1] Інформація відповідає стандарту C11 ISO/IEC 9899:2011 [7].

Ключові слова мови С

Мова містить наступні ключові слова, що є зарезервованими і не можуть бути використані у будь-якій іншій ролі (враховується регістр):

auto	**long**	**_Atomic**
break	**register**	**_Bool**
case	**restrict**	**_Complex**
char	**return**	**_Generic**
const	**short**	**_Imaginary**
continue	**signed**	**_Noreturn**
default	**sizeof**	**_Static_assert**
do	**static**	**_Thread_local**
double	**struct**	
else	**switch**	
enum	**typedef**	
extern	**union**	
float	**unsigned**	
for	**void**	
goto	**volatile**	
if	**while**	
inline	**_Alignas**	
int	**_Alignof**	

Ідентифікатори мови С

Ідентифікатори використовуються для визначення змінних, функцій, структур, об'єднань, їх полів, перечислень, власних типів даних, міток, назв макросів та їх параметрів. Один і той ж ідентифікатор може визначати різні програмні об'єкти у різних ділянках програми в залежності від області його видимості (*scope of identifier visibility*).

Правила запису ідентифікаторів:

1. Ідентифікатори складаються з букв, цифр і символу підкреслення (“**_**”) (до складу ідентифікатора не може входити будь-який спеціальний символ).
2. Першим символом повинна бути буква або символ підкреслення. Однак, не рекомендується починати ідентифікатори із символу підкреслення оскільки багато змінних стандартних бібліотечних програм починається саме з цього символу.
3. Враховується регістр. Не можна плутати в ідентифікаторах великі і малі букви (**X** і **x** – це два різні ідентифікатори).
4. Немає обмежень на максимальну довжину ідентифікатора (однак, це залежить від конкретного компілятора і згідно стандарту максимальна довжина повинна становити хоча б 31 символ).

Константи мови С

Константи поділяються на:

- цілочисельні (*integer*)[1]:
  - десяткові:
    починаються з цифри, але не нуля;
    можуть містити символи:
    0 1 2 3 4 5 6 7 8 9
    наприклад: **1024**
  - вісімкові:
    починаються з префіксу нуль "**0**";
    можуть містити символи:
    0 1 2 3 4 5 6 7
    наприклад: **037**
  - шістнадцяткові:
    починаються з префіксу "**0x**" або "**0X**";
    можуть містити символи:
    0 1 2 3 4 5 6 7 8 9
    A B C D F a b c d f
    наприклад: **0x1F**

  можуть містити суфікс:
  - "**u**" або "**U**" для явного визначення без знакових констант (**unsigned**), наприклад: **2048u**
  - "**l**" або "**L**" для явного розширення типу константи до більшого (**long**),
    наприклад: **0xAFl**
  - "**ll**" або "**LL**"для явного подвійного розширення типу (**long long**),
    наприклад: **0xAFll**

  можуть містити одночасно комбінацію суфіксів:
  - "**u**" або "**U**" та "**l**" або "**L**", порядок немає значення,
    наприклад: **025ul**
  - "**u**" або "**U**" та "**ll**" або "**LL**", порядок немає значення,
    наприклад: **256ull**
- з плаваючою комою (*floating*)[2]:

---

[1] Крім десяткових, вісімкових та шістнадцяткових існує можливість декларувати двійкові константи за допомогою префіксу "**0b**" або "**0B**". Хоч ця можливість підтримується майже всіма компіляторами, однак вона не прописана в стандарті мови.

[2] В англомовному світі та в мові С зокрема плаваючою є крапка а не кома.

числа з плаваючою комою використовуються для наближення дробових та дійсних чисел. Вони містять значущу частину або мантису (*significant part*) після якої можуть слідувати екпоненціальна частина або порядок (*exponent part*) та суфікс, що явно визначає тип;

значуща частина зазвичай містить цілу та дробну частини, що записуються цифрами і розділені крапкою "**.**";

екпоненціальна частина починається одним з символів[1] "**e**", "**E**", після яких може слідувати знак "**+**" або "**-**" та обов'язково послідовність цифр;

для правильного запису повинна бути обов'язково наявна крапка, або екпоненціальна частина, наприклад:

```
1.5
02.
.025
1e-5
.8e3
3.e+6
```

числа з плаваючою комою можуть бути шістнадцятковими. Для цього використовується один з префіксів "**0x**", або "**0X**", а експоненціальна частина починається одним з символів "**p**", "**P**". При цьому значуща частина (але не експоненціальна!) може містити шістнадцяткові символи (**A-F** або **a-f**);

наприклад: **0x1E.2p3**

за замовчуванням тип константи з плаваючою комою без суфіксу є розширеним (**double**);

для явного вказання скороченого типу (**float**) використовуються суфікси "**f**", або "**F**" наприклад: **.1e4f**

для явного вказання подвійного розширеного типу (**long double**) використовуються суфікси "**l**", або "**L**";

наприклад: **.1e4l**

числа з плаваючою комою завжди є наближеними, при компіляції вони трансформуються у власний внутрішній формат (що відповідає стандарту IEEE 754 [9]). Усі константи, що записані в однаковій початковій формі обов'язково трансформуються в однаковий внутрішній формат, наприклад: **1.23**, **1.230**, **123e-2**, **123e-02**, та **1.23L** мають різну початкову форму і тому

---

[1] Для представлення чисел з плаваючою комою дуже часто використовується науковий формат. Наприклад число $1{,}5\times10^{-5}$ записується як **1.5e-5**, або **1.5E-5**.

внутрішнє представлення у них може відрізнятися. Це необхідно враховувати в операціях порівняння, ці числа не будуть рівними;

- перечислення (*enumeration*):
  використовуються для позначення елементів перечислення;
  для їх оголошення використовується ідентифікатор;
  ідентифікатор оголошений як перечислення завжди має цілочисельний тип (**`int`**), наприклад:

```
enum colors {RED, YELLOW, GREEN, BLUE};
```

  тут у фігурних дужках ідентифікатори **`RED`**, **`YELLOW`**, **`GREEN`** та **`BLUE`** є константами перечислення, їх можна використовувати в коді програми для змістовного позначення змінних типу **`colors`**, але фактично це числа типу **`int`**.
- символьні (*character*):
  записуються у одинарних лапках, наприклад: **`'a'`**
  якщо у символьній константі присутні кілька символів, наприклад **`'ab'`**, то поведінка програми не визначається стандартом, а залежить від конкретного компілятора;
  у лапках може бути записаний будь-який символ базового алфавіту крім одинарної лапки "**`'`**", зворотної нахиленої риски "**`\`**" та символу переходу на новий рядок;
  символьними константами можуть бути керуючі послідовності (*escape-sequence*):

послідовність	шістнадцяткове значення	призначення
`\'`	**`0x27`**	записати одинарні лапки "**`'`**"
`\"`	**`0x22`**	записати подвійні лапки "**`"`**"
`\?`	**`0x3f`**	записати знак запитання "**`?`**"
`\\`	**`0x5c`**	записати зворотну нахилену риску "**`\`**"
`\a`	**`0x07`**	сигнал динаміка
`\b`	**`0x08`**	backspace
`\f`	**`0x0c`**	розрив сторінки
`\n`	**`0x0a`**	новий рядок
`\r`	**`0x0d`**	повернення каретки
`\t`	**`0x09`**	табулювання
`\v`	**`0x0b`**	вертикальне табулювання

  у якості символьної константи можна явно вказати вісімковий код, наприклад: **`'\37'`**
  або шістнадцятковий код, наприклад: **`'\xAA'`**
  за замовчуванням тип символьної константи без префіксу завжди є

без знаковим символьним (**unsigned char**);

для явного розширення типу можуть використовуватися префікси "**L**", "**u**" або "**U**", наприклад у випадку використання Unicode.

Типи і розміри даних мови С

У мові С дані поділяються на 2 групи:

- складні або структуровані.
- прості або скалярні:

  для скалярних даних існують такі базові цілочисельні типи даних:

  - **char** – найменший цілочисельний тип даних, що може вмістити в собі окремі символи базового алфавіту. Згідно стандарту вміщає числа в діапазоні хоча б **[-127,127]**. Зазвичай це один байт;
  - **int** – цілочисельний тип, що згідно стандарту вміщає числа в діапазоні хоча б **[-32767, +32767]**, але зазвичай в сучасних 32-х розрядних компіляторах займає чотири байти;

  для скалярних даних існують такі базові типи даних з плаваючою комою:

  - **float** – число з плаваючою комою одиничної точності;
  - **double** – число з плаваючою комою подвійної точності.

  робота з ними регламентується стандартом IEEE 754 [9]:

тип	байт	значущих біт	екпоненціальних біт	діапазон експоненти
float	**4**	**24**	**8**	**[-126, +127]**
double	**8**	**53**	**11**	**[-1022, +1023]**

  - для розширення базових типів використовуються кваліфікатори **short** та **long**;

Допустимі варіанти запису типу	мінімальний діапазон згідно стандарту	зазвичай байт
short **short int**	**[-32767, +32767]**	**2**
long **long int**	**[-2147483647L, +2147483647L]**	**4**
long long **long long int**	**[-9223372036854775807LL, +9223372036854775807LL]**	**8**

  кваліфікатор **long** може розширювати тип **double**, однак це розширення не регламентується стандартом IEEE 754, тому діапазони значень і розміри типу залежать від компілятора;

  для розширення базових типів використовуються кваліфікатори:

  - **signed** – явне вказання знакового типу;

- **unsigned** – явне вказання без знакового типу.

наприклад:

```
signed char
unsigned char
signed short
unsigned short int
unsigned long long
```

як випливає з означень, розмір базових типів і їх фактичні діапазони не регламентуються стандартом і залежать від конкретного компілятора, тому для визначення розміру базового типу завжди необхідно використовувати стандартну функцію часу компіляції **sizeof()**, а за необхідності визначення діапазонів конкретних типів – бібліотеку **<limits.h>**;

- довгий час у мові С не було логічного або булевського типу, хоча логічні операції використовувалися. Треба було запам'ятати, що значенню "істина" відповідає "не нуль", тобто будь-яке число, що не дорівнює нулю, а "не істина" – "нуль". З часом ситуація змінилася і в новому стандарті з'явився базовий логічний тип **_Bool**. Додатково в бібліотеці **<stdbool.h>** визначений більш звичний тип **bool**.
- **void** (*порожнеча*) – спеціальний тип, який містить пусту множину значень. Це неповний тип об'єкту, що використовується для явного вказання відсутності будь-якого типу, тому його використання відрізняється від інших типів даних. Зокрема він використовуєтья:
  - у якості типу повернення функції, якщо функція не повинна повертати значення;
  - у якості списку формальних параметрів функції, якщо функція таких немає;
  - у якості складової узагальненого вказівника, який може вказувати на будь-який об'єкт в пам'яті.

  У старих версіях компіляторів мови С цей тип був відсутній (з'явився зі стандартом С89). Якщо необхідно було зазначити, що функція не повертає результату, або немає формальних параметрів тип та параметри просто опускали. При цьому функція неявно повертала тип **int**. Узагальнені вказівники приводилися до типу вказівника на **char**. Неможливо оголосити змінну типу **void** та використовувати її у виразах, однак можна оперувати з вказівниками типу **void**.

Декларації

Всі змінні в програмі С повинні бути оголошені (задекларовані) до того як будуть використані. Деякі декларації можуть бути неявними. Декларація вказує тип і містить список одної або кількох змінних (ідентифікаторів):

```
int i, k, ms[20];
char C, cl;
```

Одночасно з декларацією змінних можлива і їх ініціалізація[1] (задання початкового значення за допомогою оператора присвоювання "**=**"[2]):

```
char ls = '0';
int k, l, lk[20], i = 1;
float eps = 1.0e-5;
```

До будь-якої змінної в декларації можна використати кваліфікатор **const**, тоді змінна перетворюється в константу, тобто її значення при виконанні програми не повинно змінюватися:

```
const int n = 20;
const double l = 2.71828182;
```

2.2.2.Структура програми

Програма на мові С умовно складається з двох частин:

- заголовка,
- тіла програми.

Заголовок складається з директив препроцесора і заголовків функцій. **Тіло програми** або функції являє собою набір операторів і міститься в фігурних дужках "**{}**". Ознакою закінчення оператора є символ крапка з комою "**;**". В одному рядку може бути кілька операторів, але бажано дотримуватись правила "одна стрічка – один оператор": програма більш наглядна і легше читається.

Коментарі в мові С обмежуються символами "**/***" і "***/**". Є можливість коментувати окремо рядки програми з допомогою символів "**//**".

Наведемо приклад простенької програми за допомогою якої вводимо з

---

[1] Згідно стандарту, явно непроініціалізовані змінні будуть містити невизначені значення, окрім випадку, якщо це статичні змінні.

[2] Оператор присвоювання "**=**" необхідний для збереження будь-яких значень в комірках пам'яті. Він є бінарним оператором, перший аргумент якого є *l-заченням* (*left*), *що модифікується* (*modifiable l-value*), тобто ідентифікатором, що посилається на деякий адрес в пам'яті – змінна, елемент масиву, поле структури даних, вказівник, адреса будь-якого з цих об'єктів. Другим аргументом є *r-значення* (*right, r-value*), тобто деякий об'єкт, що може бути присвоєний *l-заченню*, але не обов'язково є *l-заченням* – константа, вираз, результат виклику функції тощо. Наприклад, вираз **int a = 5**; тут **a** це *l-зачення*, що модифікується, а **5** – *r-значення*. Тому, вираз **5 = a;** є неправильним і при спробі його компіляції компілятор видасть повідомлення про помилку. В стандарті С11 терміни "*l-зачення, що модифікується*" та "*r-значення*" були замінені на "*значення локатора об'єкту*" та "*значення виразу*" відповідно.

клавіатури символ, а на екран виводимо код цього символу:

```
# include <stdio.h> /* header */
int main()
{
    char ch;
    printf("Enter symbol\n"); // body
    scanf("%c", &ch);
    printf("\nSymbol code %c:%d\n",ch,ch);
    return 0;
}
```

Кожна програма на мові С складається з функцій. В наведеному прикладі це функція **main()** – основна програма. Пусті дужки при імені функції означають, що ніяких вхідних параметрів основна програма не потребує. Стрічка **#include <stdio.h>** вказує компілятору, що необхідно включити інформацію, яка міститься у файлі **stdio.h** – стандартній бібліотеці вводу-виводу. Значок **#** означає, що це директива препроцесора, тобто такі команди повинні обробитися тільки один раз до процесу компіляції. Значки "**<>**" означають, що це стандартні файли, якими комплектується компілятор С.

В наведеному прикладі функція **main** складається з 4 операторів.

char ch; – оголошення змінної **ch** (ідентифікатор), типу **char**.

printf("Enter symbol\n"); – виклик бібліотечної функції виводу на екран. В дужках задається список виводу. В даному випадку виводиться стрічка символів, що міститься в подвійних лапках: **Enter symbol**

\n – керуюча послідовність, що викликає перехід на наступну стрічку.

scanf("%c", &ch); – виклик бібліотечної функції форматного вводу.

Аргументами цієї функції є:

1) специфікатор формату: **%c** ;

2) вказівник на змінну **ch** – **&ch**.

Слід пам'ятати, що для того щоб ввести за допомогою функції **scanf** якесь значення і присвоїти його змінній одного з основних типів, перед іменем змінної необхідно записати символ "**&**".

Специфікатор формату відображає тип змінної, що виводиться на друк або вводиться з клавіатури. Розрізняють такі специфікатори формату:

%d – десяткове ціле число:

%f – число з плаваючою комою, десятковий запис;

%e – число з плаваючою комою, експоненціальний запис;

%g – число з плаваючою комою або десятковий або експоненціальний запис. Використовується тільки при виводі змінних;

%c – один символ;

%s – стрічка символів;

%u – десяткове ціле без знаку;

%o – вісімкове ціле число без знаку;

%x – шістнадцяткове ціле число без знаку;

допустиме використання префіксів для розширення чи звуження типів, наприклад **%lf** – для **double**.

Четвертий оператор тіла програми **printf("\nSymbol code %c:%d\n",ch,ch);** виведе на екран повідомлення "**Symbol code**", символ, який був введений з клавіатури, а тоді ціле число, що є кодом ASCII цього символа.

Операції

Умовно операції в мові С можна розбити на такі групи:

1) Арифметичні операції:
   a. унарні: **+**;**-**;
   b. бінарні **+**;**-**;*****;**/**; (додавання, віднімання, множення, ділення).

   До операції ділення в мові С потрібно відноситись дуже уважно. Якщо обидва операнди цілого типу, то і результат буде цілого типу. Наприклад **S=2/5;** – в результаті виконання цього оператора, змінній **S** присвоїться значення **0**, щоб одержати правильний результат необхідно щоб хоча б один операнд був дійсного типу, тобто **S = 2.0/5.0**;

   У мові С є ще одна бінарна операція **%** - знаходження залишку від ділення цілих чисел. Наприклад **K=7%2;** – присвоїти змінній цілого типу **K** значення **1**, оскільки **7:2=3** і **1** в залишку. До змінних дійсного типу ця операція не застосовується.
2) Операції порівняння: **>**; **>=**; **<**; **<=**;
3) Операційні рівності: **==** – рівне; **!=** – не рівне
4) Логічні операції: **!** – логічне "ні"; **||** – логічне "або"; **&&** – логічне "і".
5) Інкрементні та декрементні операції.

   Інкрементна операція **++** додає 1 до свого операнда.

   Оператор **n++;** можна записати **n=n+1**;

   Декриментна операція **--** віднімає 1 від свого операнда.

   Розрізняють два види цих операцій:
   a. префіксні **++n** – змінна n збільшується на 1 до того, як використовується у виразі;
   b. постфіксні **n++** – змінна n збільшується на 1 після того, як її значення буде використано у виразі.

Для ілюстрації цих операцій виконаємо таку програму:

```
#include <stdio.h>
int main()
{   int a = 1, b = 1, aplus, plusb;
    aplus = a++;
    plusb = ++b;
    printf("  a aplus  b  plusb\n");
    printf("%3d%6d%3d%7d\n", a, aplus, b, plusb);
    return 0;
}
```

В результаті виконання цієї програми одержимо:

```
a aplus b plusb
2     1 2     2
```

Значення **a** збільшилось на 1 після того як виконалась операція присвоєння. Значення **b** спочатку збільшилось на 1, а тоді виконалась операція присвоєння.

6) Побітові операції

В мові С існує 6 операцій для роботи з бітами:

& – побітове “і”;

| – побітове “або”;

^ – побітове “виключне “або” (XOR);

~ – побітове “ні”;

>> – зсув вправо;

<< – зсув вліво.

& – *побітове “і”*. бінарна операція, що по розрядах порівнює два двійкові числа. Результат дорівнює 1, якщо обидва операнди рівні 1 у цьому розряді, тобто:

```
  1 0 0 1 0 0 1 1
& 0 0 1 1 1 1 0 1
-----------------
  0 0 0 1 0 0 0 1
```

| – *побітове “або”*. Результат 1, якщо хоча б у одного операнда у цьому розряді 1, тобто:

```
  1 0 0 1 0 0 1 1
| 0 0 1 1 1 1 0 1
-----------------
  1 0 1 1 1 1 1 1
```

^ – *побітове “виключне “або”*. Для кожного розряду результат дорівнює 1, якщо один з двох відповідних розрядів дорівнює 1, але не обидва одночасно:

```
  1 0 0 1 0 0 1 1
^ 0 0 1 1 1 1 0 1
-----------------
  1 0 1 0 1 1 1 0
```

~ – *побітове "ні".* Унарна операція, яка заміняє кожну 1 на 0, а 0 на 1:

```
~0b10010011 == 0b01101100
```

>> – *зсув вправо*. Зсуває розряди лівого, операнда вправо на кількість позицій вказаних у правому операнді:

```
0b10010011 << 2 == 0b00100100
```

<< - зсув вліво - зсуває розряди лівого операнда вліво на кількість розрядів, що вказані в правому операнді:

```
0b10010011 >> 2 == 0b01001100
```

позиції, які звільняються заповнюються нулями.

7) Тринарний оператор

Умовний оператор **if** у мові С можна замінити операцією виду "**?**":"

```
z=(a<b)?a:b;
```

Цей оператор відповідає оператору умовного переходу такого виду:

```
if(a<b)
    z=a;
else
    z=b;
```

8) Операція присвоєння.

Операція присвоювання може мати такий вигляд:

```
<змінна> = <вираз>;
<змінна> <знак операції>= <вираз>;
```

наприклад:

```
S=S+4;
S+=4;
```

В операції присвоювання можуть використовуватись такі операції:

+,-,*,/,%,<<,>>,&,|

Пріоритет і порядок виконання операцій

Пріоритет	Операції	Позначення	Порядок виконання
1	Виклик функції або вибір	`() [] -> .`	зліва-направо
2	Унарні операції	`+ -- ! & * ~`	справа-наліво
3	Мультиплікативні	`* / %`	зліва-направо
4	Адддитивні	`+ -`	зліва-направо
5	Зсуву	`>> <<`	зліва-направо
6	Порівняння	`> >= < <=`	зліва-направо
7	Рівності	`== !=`	зліва-направо
8	Побітове "і"	`&`	зліва-направо
9	Побітове виключне "або"	`^`	зліва-направо
10	Побітове "або"	`|`	зліва-направо
11	Логічне "і"	`&&`	зліва-направо

12	Логічне “або”	`||`	зліва-направо
13	Умови	`?:`	справа-наліво
14	Присвоювання	`= <знак>=`	справа-наліво
15	Кома	`,`	зліва-направо

Стандартні математичні функції бібліотеки **<math.h>**

Ім’я функції	Математичний запис	Тип і межі зміни аргументів	Тип результату
sin(x)	sin x	**double**	**double**
cos(x)	cos x	**double**	**double**
tan(x)	tg x	**double**	**double**
asin(x)	arcsin x	**double** $x\in[-1,1]$	$[-\pi/2,\pi/2]$
acos(x)	arccos x	**double** $x\in[-1,1]$	$[0,\pi]$
atan(x)	arctg x	**double** $x\in[-\pi/2,\pi/2]$	**double**
sinh(x)	sh x	**double**	**double**
cosh(x)	ch x	**double**	**double**
tanh(x)	th x	**double**	**double**
exp(x)	e^x	**double**	**double**
log(x)	ln x	$x>0$	**double**
log10(x)	lg x	$x>0$	**double**
pow(x,y)	x^y	**double**	**double**
sqrt(x)	$\sqrt{x}$	$x\geq 0$	**double**
fabs(x)	$\lvert x\rvert$	**double**	**double**
ldexp(x,n)	$x \cdot 2^n$	x-**double**, n-**int**	**double**
fmod(x,y)	Залишок від ділення дійсних чисел x на y	**double**	**double**

2.2.3.Перетворення типів

В операторах і виразах бажано використовувати змінні і константи однакового типу. Якщо у виразі є змішування типів компілятор автоматично перетворить типи за такими правилами. Якщо операція виконується над змінними різних типів, то обидві змінні переводяться до “вищого” з двох типів.

Порядок типів від “вищого” до “нижчого” має такий вигляд:

1) **double**;
2) **float**;
3) **long**;
д) **int**;
5) **short**;
6) **char**.

В операторах присвоювання результат перетворюється до типу змінної, якій присвоюється це значення. Це може бути як “підвищення” типу так і “пониження”, наприклад в програмі:

```
#include<stdio.h>
int main()
{
    int n, m;
    float s, p;
    n = 5 / 7.0;
    m = 4.8 / 9.2;
    s = 5 / 7;
    p = 4.8 / 9.2;
    printf("  n   m    s      p \n");
    printf("%3d %3d  %3.1f  %5.3f\n",n,m,s,p);
    return 0;
}
```

одержимо результат:

```
  n    m    s      p
  0    0  0.0  0.521
```

Найкраще при написанні програми уникати перетворення типів, особливо в бік “пониження”. Можливий варіант явного приведення типів

(<ім'я типу>)<змінна або константа>.

Так у прикладі:

```
m = 1.6 + 1.8;
n = (int)1.6 + (int)1.8;
```

змінній **m** присвоїться значення **3**, оскільки спочатку додадуться два дійсних числа **1.6** і **1.8** від результату **3.4** відкинеться дробова частина. Змінній **n** присвоїться значення **2**, оскільки **(int)1.6==1**, **(int)1.8==1**; **1+1==2**.

2.2.4.Оператори

Будь-яка програма складається з послідовності операторів. Ознакою закінчення оператора є крапка з комою “**;**”. Так запис **S=5** не є оператором, це просто вираз, а **S=5;** це вже оператор присвоювання.

Блок – це група операторів, що міститься у фігурних дужках, вони використовуються:

- щоб згрупувати кілька логічно зв’язаних операторів в один;
- як тіло функції;
- для локалізації дії описів (визначають область видимості ідентифікаторів).

Оператор **if**

Оператор **if-else** використовується для вибору одного з двох варіантів рішення. Синтаксичний опис оператора **if-else**:

```
if(<вираз>)
    <оператор 1>;
else
    <оператор 2>;
```

Обчислюється "**вираз**", якщо його значення "істина" (тобто не нуль) виконується "**оператор 1**", якщо "не істина" (тобто нуль) виконується "**оператор 2**". Частина **else** може бути відсутня. При вкладених **if-else** необхідно пам'ятати, що **else** відноситься до внутрішнього **if**:

```
if(x>0)
    if(a>b)
        z = a;
    else
        z = b;
```

Якщо треба змінити порядок необхідно використати фігурні дужки, тобто виділити блок:

```
if(x>0)
{
    if(a>b) z = a;
}
else z = b;
```

Щоб розгалузити програму можна використати конструкцію: **else if**:

```
if(<вираз 1>)
    <оператор 1>;
else if(<вираз 2>)
    <оператор 2>;
else if(<вираз 3>)
    <оператор 3>;
else
    <оператор 4>;
```

Якщо "**вираз 1**" – "істина" виконується "**оператор 1**", якщо ні – перевіряється "**вираз 2**". Якщо "**вираз 2**" – "істина", виконується "**оператор 2**" і так далі.

Оператор **switch**

Оператор **switch** (перемикач) використовується для вибору одного з багатьох варіантів.

Синтаксис оператора **switch**:

```
switch(<вираз>)
{
case <константа вибору 1>:<оператор 1>;
case <константа вибору 2>:<оператор 2>;
…
default: <оператор n>
}
```

Якщо “вираз” співпадає з одною із констант вибору, то виконується відповідний оператор або блок операторів. Якщо “вираз” не співпадає з жодною з констант вибору – виконується оператор після слова “**default**”. Для прикладу промоделюємо роботу світлофора:

```
#include<stdio.h>
int main()
{
    char ch;
    printf("Enter first letter of color\n");
    scanf("%c",&ch);
    switch(ch)
    {
    case 'r': printf("Wait!\n"); break;
    case 'y': printf("Attention\n"); break;
    case 'g': printf("Go!\n"); break;
    default : printf("There is no such color!\n");
              break;
    }
    return 0;
}
```

“Вираз” і константи вибору повинні бути цілого типу, або типу **char**. Заборонено використовувати в якості константи вибору змінну. Оператор **break** здійснює негайний вихід з оператора **switch**. Якщо цього оператора немає, то будуть виконані оператори всіх варіантів після вибраного.

2.3. Контрольні запитання

1. Правила запису ідентифікаторів.
2. Які Ви знаєте типи даних і як вони описуються?
3. Структура програми на мові С.
4. Що таке побітові операції?
5. Що таке оператор, ознака закінчення оператора.
6. Як здійснюється узгодження типів у виразах?
7. Яка різниця між оператором і блоком у мові С?
8. Синтаксис умовного оператора **`if`**.
9. Напишіть приклад оператора **`switch`**.
10. Для чого використовується оператор **`break`**?

2.4. Практичне завдання

1. Вивчити основні поняття мови програмування С, операції, стандартні функції, оператори розгалуження програм.
2. Написати програму калькулятора.
3. Побудувати блок-схеми алгоритмів відповідно до завдання.
4. Скласти програми на алгоритмічній мові С згідно завдання.
5. Відлагодити програму, виконати обчислення, проаналізувати отримані результати.

2.5. Зміст звіту

1. Титульний аркуш.
2. Мета роботи.
3. Практичне завдання.
4. Блок-схема алгоритмів у відповідності до завдання.
5. Тексти програм у відповідності до завдання.
6. Результати обчислень.
7. Аналіз результатів, висновки.

2.6. Практичне завдання

Написати програму простого калькулятора з вводом даних з клавіатури і виводом результату на екран. Кожен може обрати собі одну або декілька математичних операцій або алгебраїчних функцій.