



ОПЕРАТОРИ МОВИ ПРОГРАМУВАННЯ С

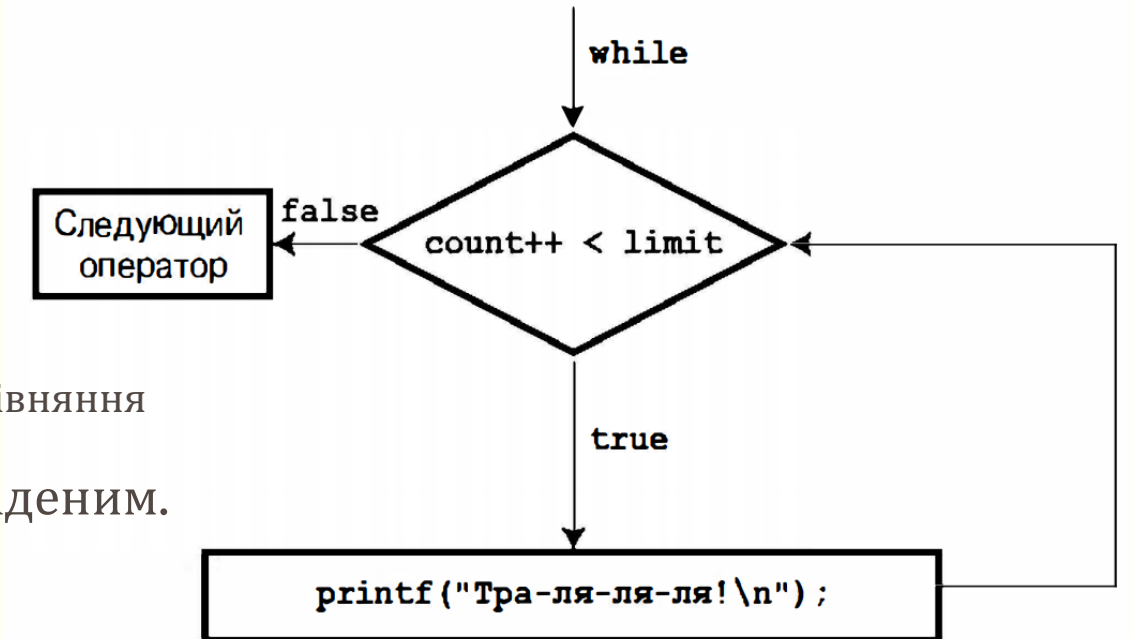
Питання 2.3.

Оператори мови програмування

- *Оператори циклу (loop statements)*
 - while
 - do-while
 - for
- *Оператори умовного переходу (conditional statements)*
 - if-else
 - switch-case
- *Оператор безумовного переходу (jump statements)*
 - goto
 - break
 - continue
 - return

Цикл з передумовою. Оператор while

- Загальна форма оператору:
 - **while** (вираз)
оператор
 - Вираз може бути будь-яким.
 - У попередніх прикладах використовувалось порівняння
 - Оператор може бути як простим, так і складеним.
-
- Якщо вираз істинний (загалом приймає ненульове значення), оператор виконується один раз, після чого вираз перевіряється знову.
 - Кожна послідовність перевірки та виконання називається *ітерацією*.

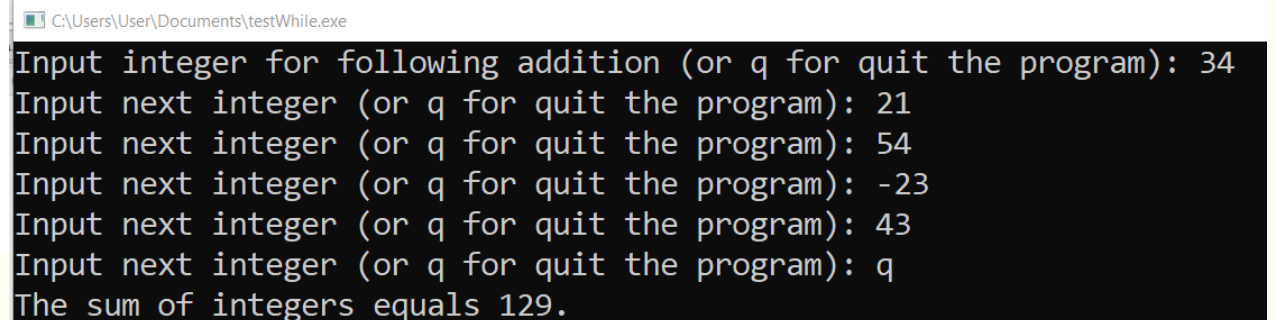


Цикл з передумовою while

```
1  /* summing.c -- sums integers entered interactively */
2  #include <stdio.h>
3  int main(void)
4  {
5      long num;
6      long sum = 0L;      /* initialize sum to zero */
7      int status;
8
9      printf("Please enter an integer to be summed ");
10     printf("(q to quit): ");
11     status = scanf("%ld", &num);
12     while (status == 1) /* == means "is equal to" */
13     {
14         sum = sum + num;
15         printf("Please enter next integer (q to quit): ");
16         status = scanf("%ld", &num);
17     }
18     printf("Those integers sum to %ld.\n", sum);
19
20     return 0;
21 }
```

Для завершення вводу даних використовується значення, яке повертає функція `scanf()`.

- Тип `long` дозволяє вводити великі числа



```
C:\Users\User\Documents\testWhile.exe
Input integer for following addition (or q for quit the program): 34
Input next integer (or q for quit the program): 21
Input next integer (or q for quit the program): 54
Input next integer (or q for quit the program): -23
Input next integer (or q for quit the program): 43
Input next integer (or q for quit the program): q
The sum of integers equals 129.
```

Більш компактний запис коду

- ```
status = scanf("%ld", &num);
while (status == 1)
{
 /* дії, що виконуються в циклі */
 status = scanf ("%ld", &num) ;
}
```
- ```
while (scanf("%ld", &num) == 1)  
{  
    /* дії, що виконуються в циклі */  
}
```

 - При успішному виклику функції `scanf()` вхідне значення поміщається у змінну `num`.
 - `scanf()` повертає 1 або 0 і не є значенням змінної `num`) управляет циклом.
- функція `scanf()` викликається на кожній ітерації, при цьому вводиться нове значення `num` та відбувається нова перевірка.

Коли цикл завершується?

```
1 // when.c -- when a loop quits
2 #include <stdio.h>
3 int main(void)
4 {
5     int n = 5;
6
7     while (n < 7)
8     {
9         printf("n = %d\n", n);
10        n++;
11        printf("Now n = %d\n", n);
12    }
13    printf("The loop has finished.\n");
14
15    return 0;
16 }
```



when.c

- Існують цикли
 - З передумовою
 - З післяумовою (постумовою)
 - З лічильником

Выбрать C:\Users\User\Documents\when.exe

```
n = 5
Now n = 6
n = 6
Now n = 7
Loop is ended.
```

Цикл while с передумовою

```
1  /* while1.c -- watch your braces */
2  /* bad coding creates an infinite loop */
3  #include <stdio.h>
4  int main(void)
5  {
6      int n = 0;
7
8      while (n < 3)
9          printf("n is %d\n", n);
10     n++;
11     printf("That's all this program does\n");
12
13     return 0;
14 }
```



while1.c

 Выбрать C:\Users\User\Documents\while1.exe

[illegible]

Позначення істини в мові C

- Вираз у мові C завжди має значення
 - Нескінченний цикл: `while (1) { }`

```
1  /* t_and_f.c -- true and false values in C */
2  #include <stdio.h>
3  int main(void)
4  {
5      int true_val, false_val;
6
7      true_val = (10 > 2);    // value of a true relationship
8      false_val = (10 == 2); // value of a false relationship
9      printf("true = %d; false = %d \n", true_val, false_val);
10
11     return 0;
12 }
```



C:\Users\User\Documents>true_false.exe

true = 1; false = 0

- Часто використовується з умовами перевірки.
 - Наприклад, конструкцію `while (goats != 0)` можна замінити на `while (goats)`.

```
1  // truth.c -- what values are true?
2  #include <stdio.h>
3  int main(void)
4  {
5      int n = 3;
6
7      while (n)
8          printf("%2d is true\n", n--);
9      printf("%2d is false\n", n);
10
11     n = -3;
12     while (n)
13         printf("%2d is true\n", n++);
14     printf("%2d is false\n", n);
15
16     return 0;
17 }
```



truth.c

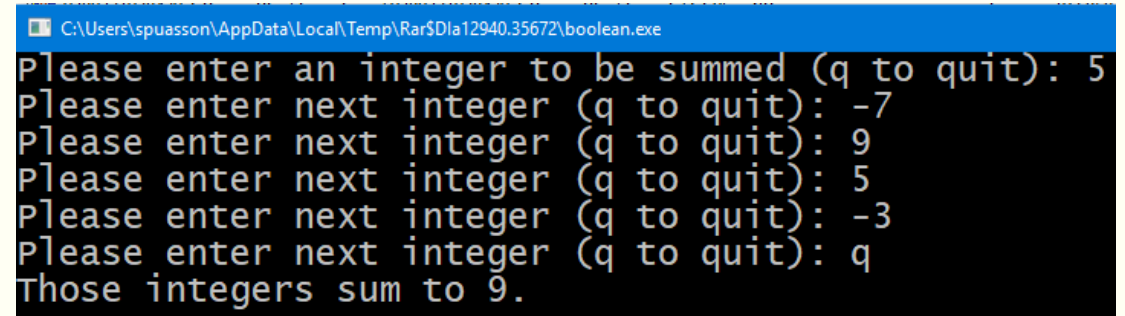
C:\Users\User\Documents\truth.exe

```
3 is true
2 is true
1 is true
0 is false
-3 is true
-2 is true
-1 is true
0 is false
```


Новий тип _Bool

```
1 // boolean.c -- using a _Bool variable
2 #include <stdio.h>
3 int main(void)
4 {
5     long num;
6     long sum = 0L;
7     _Bool input_is_good;
8
9     printf("Please enter an integer to be summed ");
10    printf("(q to quit): ");
11    input_is_good = (scanf("%ld", &num) == 1);
12    while (input_is_good)
13    {
14        sum = sum + num;
15        printf("Please enter next integer (q to quit): ");
16        input_is_good = (scanf("%ld", &num) == 1);
17    }
18    printf("Those integers sum to %ld.\n", sum);
19
20    return 0;
21 }
```

- Традиційно в мові C змінні для представлення значень true і false мають тип int.
 - Стандарт C99 вводить спеціальний тип _Bool.
 - Змінна типу _Bool може набувати тільки значення 1 (істина) і 0 (хиба).
 - При присвоєнні ненульового числового значення змінній типу _Bool, вона набуде значення 1.



```
C:\Users\spuasson\AppData\Local\Temp\Rar$Dla12940.35672\boolean.exe
Please enter an integer to be summed (q to quit): 5
Please enter next integer (q to quit): -7
Please enter next integer (q to quit): 9
Please enter next integer (q to quit): 5
Please enter next integer (q to quit): -3
Please enter next integer (q to quit): q
Those integers sum to 9.
```



boolean.c

Невизначені цикли та цикли з лічильником

- **Невизначені цикли:** заздалегідь невідомо, скільки разів буде виконано цикл.
- **Цикли з лічильником:** цикли виконують наперед задану кількість ітерацій.

```
1 // sweetie1.c -- a counting loop
2 #include <stdio.h>
3 int main(void)
4 {
5     const int NUMBER = 22;
6     int count = 1;           // initialization
7
8     while (count <= NUMBER)  // test
9     {
10         printf("Be my Valentine!\n"); // action
11         count++;                 // update count
12     }
13
14     return 0;
15 }
```



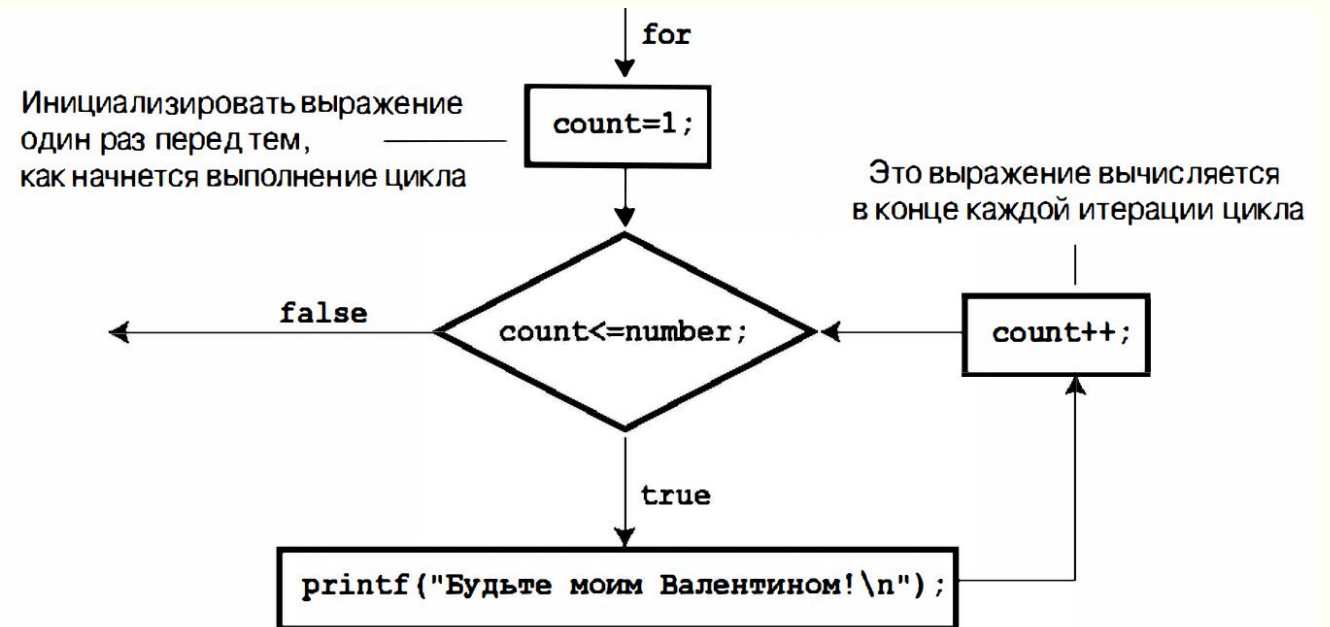
sweetie1.c

[illegible]

Цикл for

- Для організації циклу із заздалегідь заданою кількістю ітерацій слід:
 1. Ініціалізувати лічильник
 2. Порівняти значення лічильника з певною обмеженою величиною
 3. Оновити значення лічильника після кожного проходу циклу.

```
1 // sweetie2.c -- a counting loop using for
2 #include <stdio.h>
3 int main(void)
4 {
5     const int NUMBER = 22;
6     int count;
7
8     for (count = 1; count <= NUMBER; count++)
9         printf("Be my Valentine!\n");
10
11     return 0;
12 }
```



Особливості використання циклу for

- Крок по циклу може бути різним:

```
#include <stdio.h>
int main ( void) {
    int n;
    for ( n = 2; n < 60; n = n + 13 )
        printf ( "%d\n", n);
    return 0;
}
```

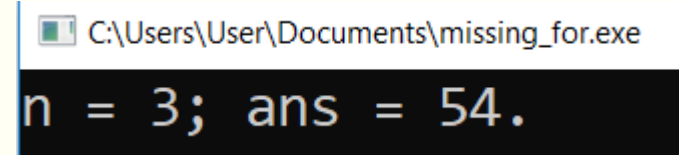
- Можна лічити по символах, а не числах:

```
#include <stdio . h>
int main (void) {
    char ch ;
    for ( ch = 'a'; ch <= 'z' ; ch++ )
        printf ( "ASCII value for %c equals %d. \n" , ch , ch) ;
    return 0;
}
```

Особливості використання циклу for

- Вирази в умові циклу можна опускати, проте обов'язково ставити порожній оператор

```
1  /* for_none.c */
2  #include <stdio.h>
3  int main(void)
4  {
5      int ans, n;
6
7      ans = 2;
8      for (n = 3; ans <= 25; )
9          ans = ans * n;
10     printf("n = %d; ans = %d.\n", n, ans);
11     return 0;
12 }
```



```
C:\Users\User\Documents\missing_for.exe
n = 3; ans = 54.
```


for_none.c

Локальні змінні та оператори МП

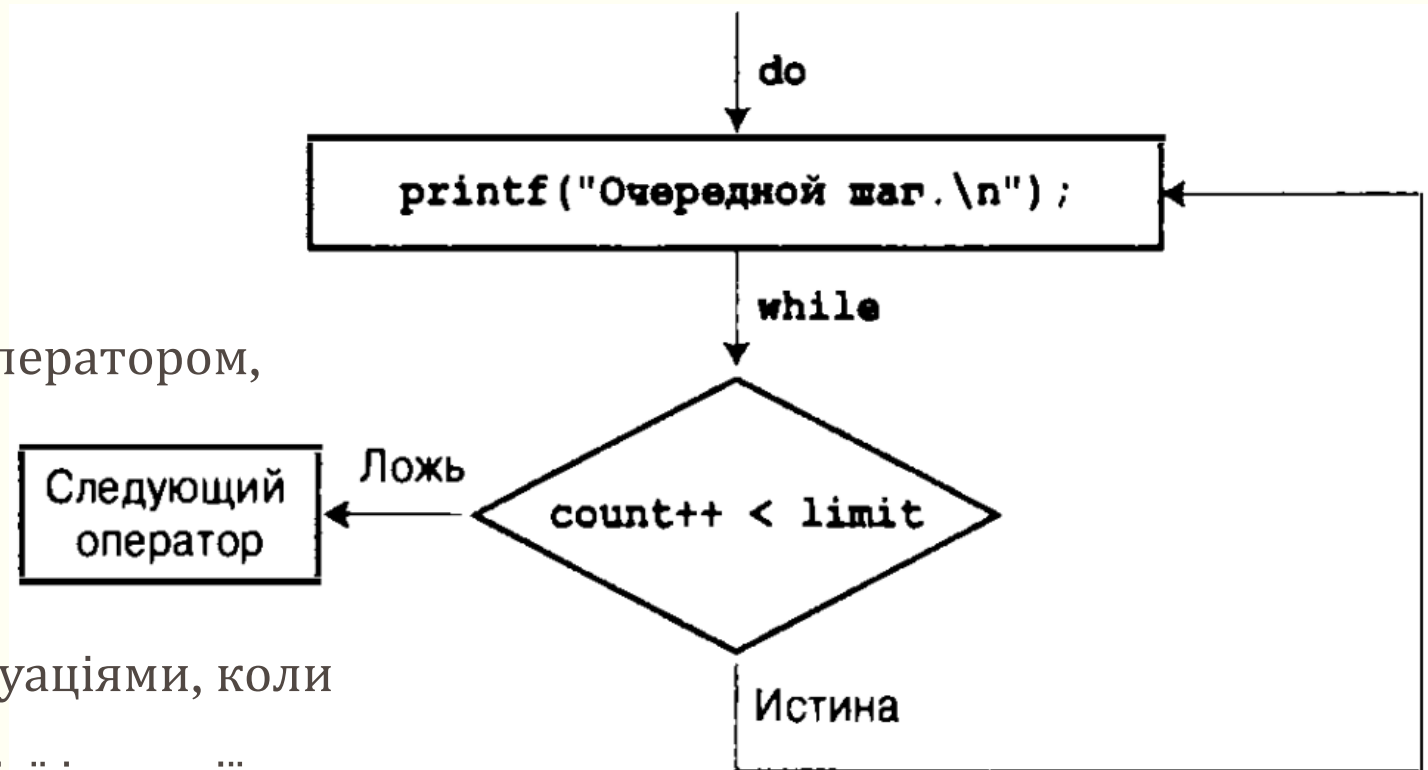
```
1  #include <stdio.h>
2
3  int main(void) {
4      int i = 0;
5      for (; i < 10; i++) {
6          int local = i;
7          printf("Twice Iteration %d + %d \n", i, local);
8      }
9      printf("Local variable i = %d reached. Unreachable local variable local = %d", i, local);
10     return 0;
11 }
```

■ Помилки компіляції:

Файл	Сообщение
E:\GDisk\[College]\[Основи програмування та алго...	In function 'main':
E:\GDisk\[College]\[Основи програмування та алгоритмічні ...	[Error] 'local' undeclared (first use in this function)
E:\GDisk\[College]\[Основи програмування та алгоритмічні ...	[Note] each undeclared identifier is reported only once for each function it appears in

Цикл з постумовою do-while

- Перевірка умови відбувається після виконання ітерації
- Загальна форма запису:
do
 оператор
while (вираз);
- Оскільки do-while вважається оператором,
 потрібна ; в кінці
- Використання обмежується ситуаціями, коли
 потрібне виконання хоча б однієї ітерації.

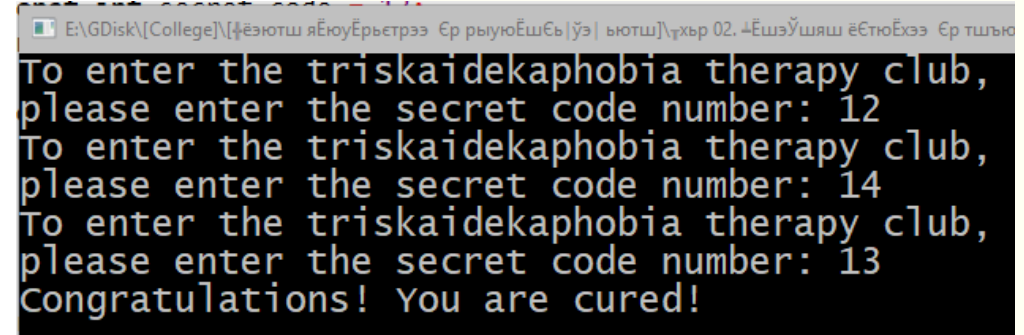


Цикл з постумовою do-while

- Приклад використання

```
do
{
    запросить ввод пароля
    прочитать пользовательский ввод
} while (введенные данные не совпадают с паролем);
```

```
1  /* do_while.c -- exit condition loop */
2  #include <stdio.h>
3  int main(void)
4  {
5      const int secret_code = 13;
6      int code_entered;
7
8      do
9      {
10         printf("To enter the triskaidekaphobia therapy club,\n");
11         printf("please enter the secret code number: ");
12         scanf("%d", &code_entered);
13     } while (code_entered != secret_code);
14     printf("Congratulations! You are cured!\n");
15
16     return 0;
17 }
```



```
E:\GDisk\[College]\[Ёэютш яЁюЁрьётрээ Ёр рыуюЁшЁь|ўэ| ьютш]\хър 02. 4Ёшэўшыш ёЁтюЁхээ Ёр тшью
To enter the triskaidekaphobia therapy club,
please enter the secret code number: 12
To enter the triskaidekaphobia therapy club,
please enter the secret code number: 14
To enter the triskaidekaphobia therapy club,
please enter the secret code number: 13
Congratulations! You are cured!
```



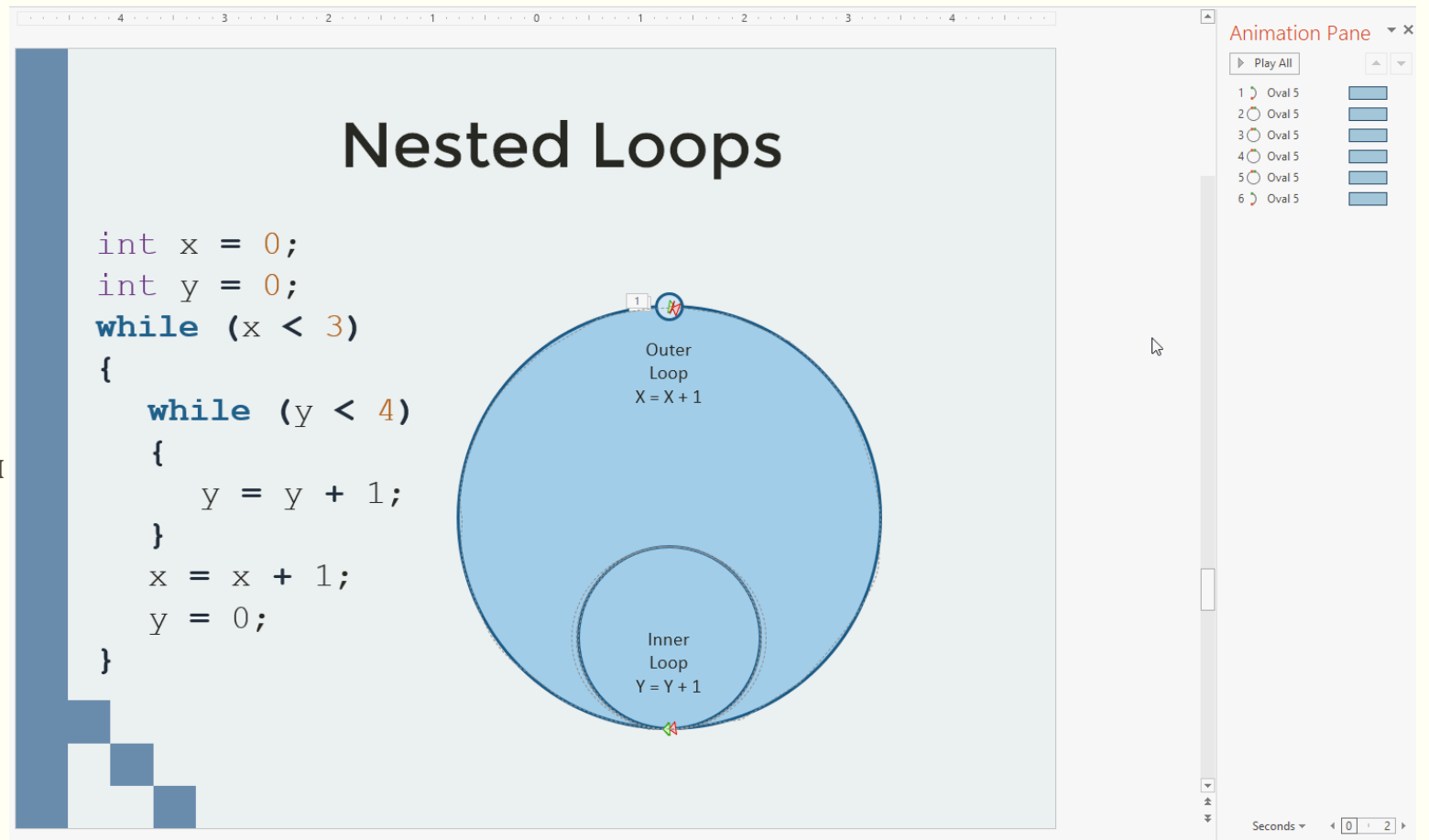
do_while.c

Вибір доречного циклу

- Зазвичай обирають цикл з передумовою.
 - Загалом краще перевіряти умову до запуску ітерації. Цикл може бути пропущеним взагалі.
 - Програма простіша для сприйняття.
- Цикл for чи while?
 - Частково це вподобання програміста.
 - Цикл for можна частково уподібнити до while, прибравши останній блок умови
 - Цикл for доречніший, коли передбачається ініціалізація та оновлення змінної
 - Цикл while краще застосовувати для довгих умов
 - `while(scanf("%ld", &num) == 1)`

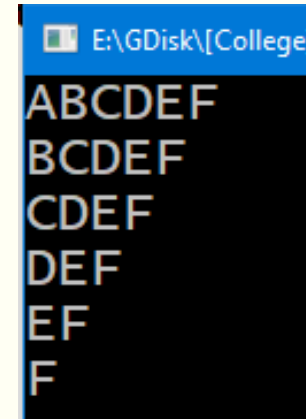
Вкладені цикли

- Цикл, який виконується всередині іншого циклу.
- Один цикл може обробляти всі стовпці в рядку, а другий – всі рядки.
 - Рядок 3 – зовнішній цикл
 - Рядок 5 – внутрішній цикл



Залежність між зовнішнім та внутрішнім циклами

```
1 // rows2.c -- using dependent nested loops
2 #include <stdio.h>
3 int main(void)
4 {
5     const int ROWS = 6;
6     const int CHARS = 6;
7     int row;
8     char ch;
9
10    for (row = 0; row < ROWS; row++)
11    {
12        for (ch = ('A' + row); ch < ('A' + CHARS); ch++)
13            printf("%c", ch);
14        printf("\n");
15    }
16
17    return 0;
18 }
```



rows2.c

Оператор if (оператор галуження, оператор вибору)

- Вузловий пункт, де програма повинна здійснити вибір. Має вигляд **if (вираз)**

оператор

- Якщо вираз істинний, умова виконується, інакше – пропускається.
- Зазвичай використовуються умови відношення (порівнюються величини)
- Тіло оператора формується аналогічно до циклу while

```
if (score > big)
    printf("Джекпот!\n");           // простой оператор
if (joe > ron)
{
    joecash++;
    printf("Ты проиграл, Ron.\n");
}
```

- Вся структура if вважається одним оператором навіть при наявності складеного оператору

Впроваджуємо оператор if

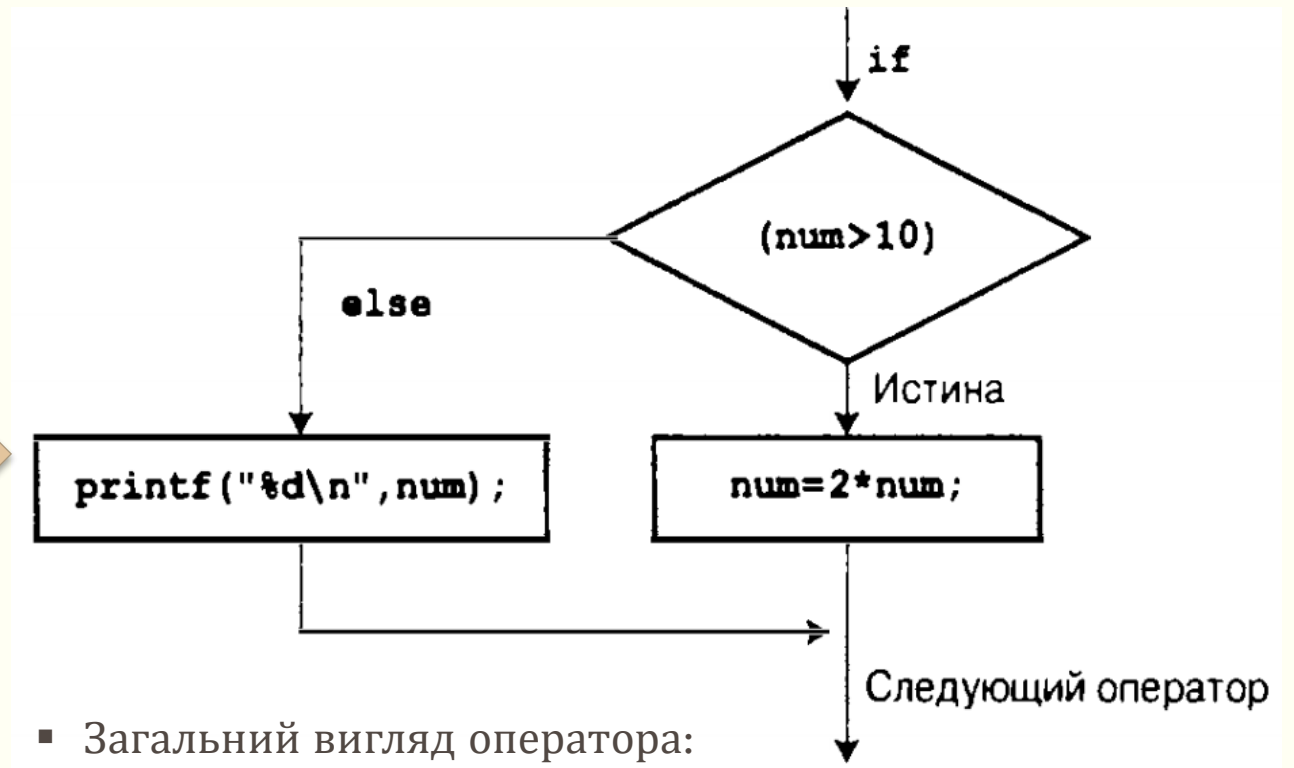
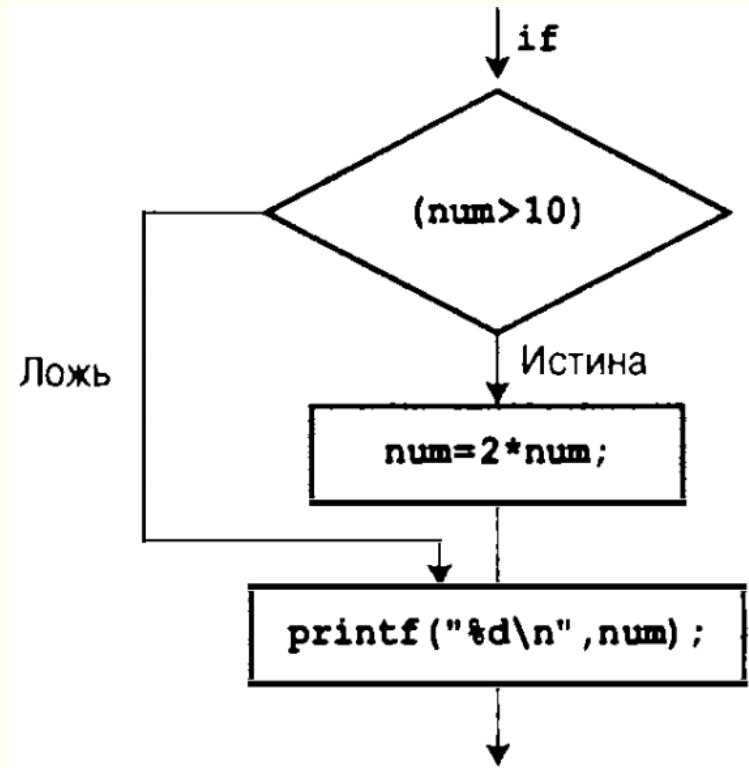
```
1 // colddays.c -- finds percentage of days below freezing
2 #include <stdio.h>
3 int main(void)
4 {
5     const int FREEZING = 0;
6     float temperature;
7     int cold_days = 0;
8     int all_days = 0;
9
10    printf("Enter the list of daily low temperatures.\n");
11    printf("Use Celsius, and enter q to quit.\n");
12    while (scanf("%f", &temperature) == 1)
13    {
14        all_days++;
15        if (temperature < FREEZING)
16            cold_days++;
17    }
18    if (all_days != 0)
19        printf("%d days total: %.1f%% were below freezing.\n",
20               all_days, 100.0 * (float) cold_days / all_days);
21    if (all_days == 0)
22        printf("No data entered!\n");
23
24    return 0;
25 }
```

```
E:\GDisk\College\{фёютш яёюуёрьстрээ ёр рыуюёшё|ёэ| ёютш}\тхёр 02. 4ёшэўшыш ёётюёхээ ё
Enter the list of daily low temperatures.
Use Celsius, and enter q to quit.
12 5 -2.5 0 6 8 -3 -10 5 10 q
10 days total: 30.0% were below freezing.
```



colddays.c

Додавання конструкції else



- Загальний вигляд оператора:

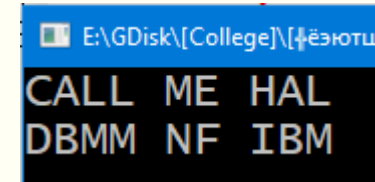
```
if (вираз)
    оператор1
else
    оператор2
```

Приклад: шифр із зсувом на 1 символ за алфавітом

```

1 // cypher1.c -- alters input, preserving spaces
2 #include <stdio.h>
3 #define SPACE ' ' // that's quote-space-quote
4 int main(void)
5 {
6     char ch;
7
8     ch = getchar(); // read a character
9     while (ch != '\n') // while not end of line
10    {
11        if (ch == SPACE) // leave the space
12            putchar(ch); // character unchanged
13        else
14            putchar(ch + 1); // change other characters
15        ch = getchar(); // get next character
16    }
17    putchar(ch); // print the newline
18
19    return 0;
20 }

```



cypher1.c

Для символного вводу-виводу передбачені спеціальні функції:

- *getchar()* – повертає наступний символ із вхідного потоку, нічого не приймає;
- *putchar()* – виводить переданий аргумент (символ) у консоль;

Визначені в `studio.h`.

- зазвичай реалізовані у вигляді макроса препроцесора
- специфікатори формату не потрібні, оскільки обробляються лише символи.

Спрощення коду

```
ch = getchar();      /* читать символ          */
while (ch != '\n')   /* пока не встретится конец строки */
{
    ...              /* обработать символ          */
    ch = getchar();   /* получить следующий символ      */
}
```



```
while ((ch = getchar()) != '\n')
{
    /* обработать символ          */
}
```

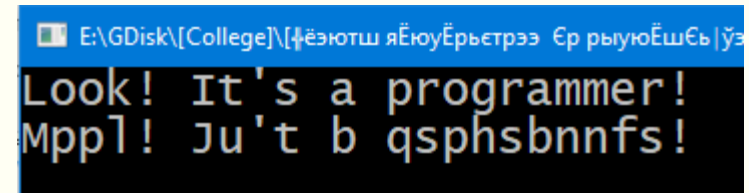
- Усі дужки обов'язкові.

Функції для роботи з символами із ctype.h

Функції	Опис
<u>isalpha()</u>	Перевіряє, чи символ є літерою
<u>isdigit()</u>	Перевіряє, чи є символ цифрою
<u>isalnum()</u>	Перевіряє, чи є символ alphanumeric (A-Z+a-z+0-9, якщо регістр неважливий)
<u>isspace()</u>	Перевіряє, чи є символ пробільним (' ', '\t', '\n', '\v', '\f', '\r', carriage return (CR))
<u>islower()</u>	Перевіряє, чи є символ малою буквою
<u>isupper()</u>	Перевіряє, чи є символ великою буквою
<u>isxdigit()</u>	Перевіряє, чи є символ шістнадцятковою цифрою
<u>iscntrl()</u>	Перевіряє, чи є символ керівним символом (Ctrl+)
<u>isprint()</u>	Перевіряє, чи є символ друкованим символом(включаючи пробіл)
<u>ispunct()</u>	Перевіряє, чи є символ пунктуаційним (! " # \$ % & ' () * +, - . / : ; ? @ [\] ^ _ ` { } ~)
<u>isgraph()</u>	Перевіряє, чи є символ друкованим (не включаючи пробіл)
<u>tolower()</u>	Перевіряє, чи є символ alphabetic та конвертує його в нижній регістр
<u>toupper()</u>	Перевіряє, чи є символ alphabetic та конвертує його в верхній регістр

Приклад використання функції isalpha()

```
1 // cypher2.c -- alters input, preserving non-letters
2 #include <stdio.h>
3 #include <ctype.h>           // for isalpha()
4 int main(void)
5 {
6     char ch;
7
8     while ((ch = getchar()) != '\n')
9     {
10         if (isalpha(ch))      // if a letter,
11             putchar(ch + 1);  // display next letter
12         else                  // otherwise,
13             putchar(ch);      // display as is
14     }
15     putchar(ch);              // display the newline
16
17     return 0;
18 }
```



E:\GDisk\[College]\[фёютш яёюуёрьстрээ ёр рыуюёшёь|ўэ
Look! It's a programmer!
Mppl! Ju't b qsphsbnnfs!



cypher2.c

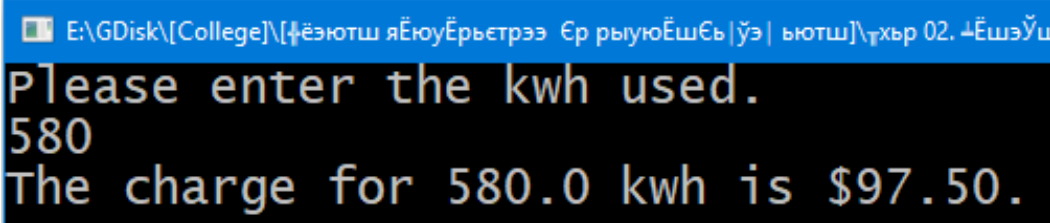
```

1 // electric.c -- calculates electric bill
2 #include <stdio.h>
3 #define RATE1 0.13230 // rate for first 360 kwh
4 #define RATE2 0.15040 // rate for next 108 kwh
5 #define RATE3 0.30025 // rate for next 252 kwh
6 #define RATE4 0.34025 // rate for over 720 kwh
7 #define BREAK1 360.0 // first breakpoint for rates
8 #define BREAK2 468.0 // second breakpoint for rates
9 #define BREAK3 720.0 // third breakpoint for rates
10 #define BASE1 (RATE1 * BREAK1)
11 // cost for 360 kwh
12 #define BASE2 (BASE1 + (RATE2 * (BREAK2 - BREAK1)))
13 // cost for 468 kwh
14 #define BASE3 (BASE1 + BASE2 + (RATE3 * (BREAK3 - BREAK2)))
15 //cost for 720 kwh
16 int main(void)
17 {
18     double kwh; // kilowatt-hours used
19     double bill; // charges
20
21     printf("Please enter the kwh used.\n");
22     scanf("%lf", &kwh); // %lf for type double
23     if (kwh <= BREAK1)
24         bill = RATE1 * kwh;
25     else if (kwh <= BREAK2) // kwh between 360 and 468
26         bill = BASE1 + (RATE2 * (kwh - BREAK1));
27     else if (kwh <= BREAK3) // kwh between 468 and 720
28         bill = BASE2 + (RATE3 * (kwh - BREAK2));
29     else // kwh above 680
30         bill = BASE3 + (RATE4 * (kwh - BREAK3));
31     printf("The charge for %.1f kwh is $%1.2f.\n", kwh, bill);
32
33     return 0;
34 }

```

Множинний вибір else if

Первые 360 кВт/ч:	\$0.13230 за 1 кВт/ч
Следующие 108 кВт/ч	\$0.15040 за 1 кВт/ч
Следующие 252 кВт/ч	\$0.30025 за 1 кВт/ч
Свыше 720 кВт/ч	\$0.34025 за 1 кВт/ч



```

E:\GDisk\[College]\[фёютш яёюуёрьетрэз ёр рыуюёшёь|ёэ| ёютш]\тхър 02. 4ёшэўц
Please enter the kwh used.
580
The charge for 580.0 kwh is $97.50.

```



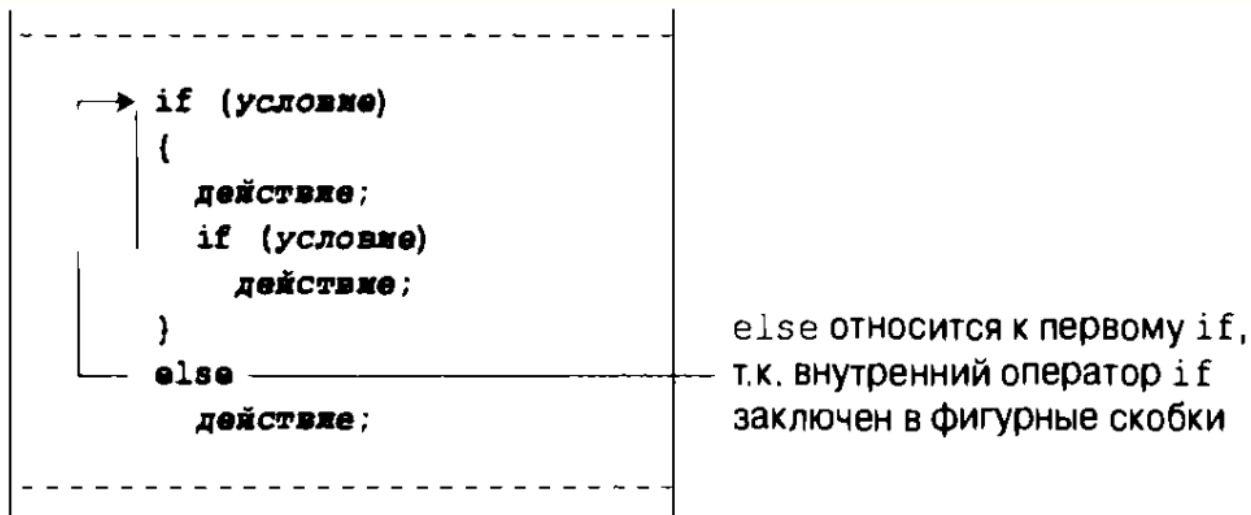
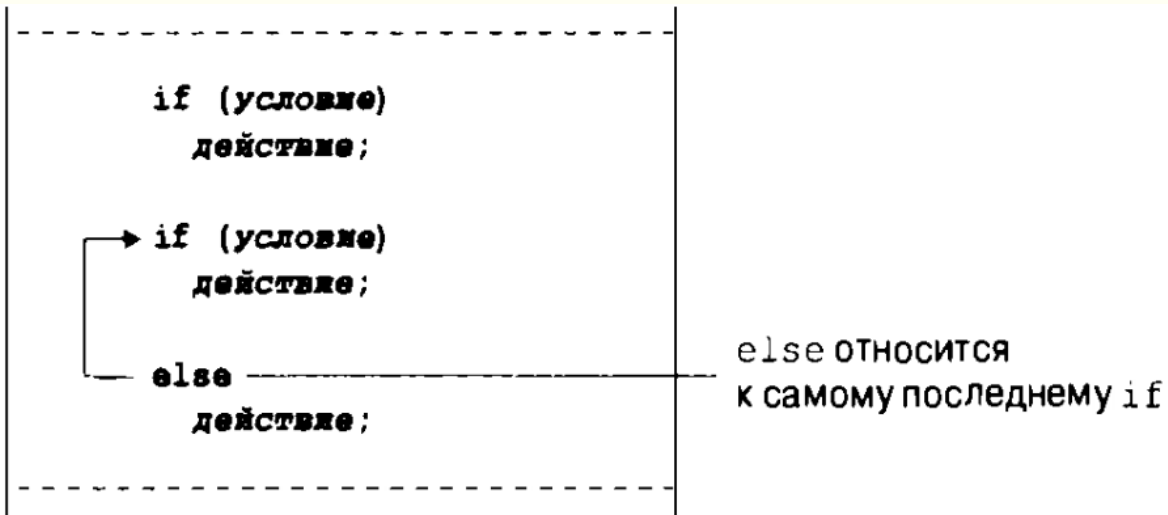
electric.c

- У стандарті C99 передбачається мінімум 127 рівнів вкладеності операторів МП

Утворення пар else та if

```
if (number > 6)
    if (number < 12)
        printf("Вы закончили игру!\n");
else
    printf("К сожалению, вы потеряли право хода!\n");
```

Якому if належить else?



Вкладання різних операторів мови програмування

■ Приклад. Пошук простих чисел.

- Ідея: перевірити всі можливі дільники.

```
for (div = 2; div < num; div++)  
    if (num % div == 0)  
        printf("%d делится на %d\n", num, div);
```

- Можна перевіряти не всі дільники, а від 2 до \sqrt{num}

```
for (div = 2; (div * div) <= num; div++)  
    if (num % div == 0)  
        printf("%d делится на %d и %d.\n",  
            num, div, num / div);
```

■ Недоліки:

- Для точних квадратів printf() виведе однакові числа – потрібна додаткова перевірка
- Механізм визначення простоти числа не реалізовано – ввести змінну-прапорець для перевірки

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35

```
E:\GDisk\College\Цезантш яЁюгрьстрээ Ёр рыуюЁшсё\ўз| ьютш\хър 02. 4Ёшэўшяш ёСтюЁхээ Ёр тшлюэрээ яЁюгрь\Code\divisors.exe
Please enter an integer for analysis; Enter q to quit.
144
144 is divisible by 2 and 72.
144 is divisible by 3 and 48.
144 is divisible by 4 and 36.
144 is divisible by 6 and 24.
144 is divisible by 8 and 18.
144 is divisible by 9 and 16.
144 is divisible by 12.
Please enter another integer for analysis; Enter q to quit.
112
112 is divisible by 2 and 56.
112 is divisible by 4 and 28.
112 is divisible by 7 and 16.
112 is divisible by 8 and 14.
Please enter another integer for analysis; Enter q to quit.
q
Bye.
```



divisors.c

Комбінування кількох умов у єдиний вираз

- Кілька умов можна комбінувати за допомогою логічних операторів: `&&`, `||`, `!`.

- Наприклад, за 100-бальною шкалою оцінка «відмінно» коливається від 90 до 100 балів.

```
if (range >= 90 && range <= 100)
    printf("Неплохой результат!\n");
```

- Так неправильно:

```
if (90 <= range <= 100)    // Не поступайте так!
    printf("Неплохой результат!\n");
```

- Помилка семантична: вираз трактується як $(90 \leq \text{range}) \leq 100$
- Умова $90 \leq \text{range}$ дає 0 або 1, які завжди менші за 100
- Загальна умова завжди істинна

- Поширений код:

```
if (ch >= 'a' && ch <= 'z')
    printf("Это строчная буква.\n");
```

Працює не для всіх кодувань

```
if (islower(ch))
    printf("Это строчный символ.\n");
```

Не залежить від кодування

Узагальнення знань: програма для підрахунку слів

```
1 // wordcnt.c -- counts characters, words, lines
2 #include <stdio.h>
3 #include <ctype.h>           // for isspace()
4 #include <stdbool.h>         // for bool, true, false
5 #define STOP '|'
6 int main(void)
7 {
8     char c;                  // read in character
9     char prev;               // previous character read
10    long n_chars = 0L;        // number of characters
11    int n_lines = 0;          // number of lines
12    int n_words = 0;          // number of words
13    int p_lines = 0;          // number of partial lines
14    bool inword = false;      // == true if c is in a word
15
16    printf("Enter text to be analyzed (| to terminate):\n");
17    prev = '\n';              // used to identify complete lines
18    while ((c = getchar()) != STOP)
19    {
20        n_chars++;             // count characters
21        if (c == '\n')
22            n_lines++;         // count lines
23        if (!isspace(c) && !inword)
24        {
25            inword = true;     // starting a new word
26            n_words++;         // count word
27        }
28        if (isspace(c) && inword)
29            inword = false;    // reached end of word
30        prev = c;              // save character value
31    }
```

читать символ

пока еще имеются входные данные

инкрементировать счетчик символов

если строка прочитана, инкрементировать счетчик строк

если слово прочитано, инкрементировать счетчик слов

читать следующий символ

- STOP – умовний символ кінця рядка

- Пізніше дізнаємось про '\0'

- Словом вважаємо неперервну непусту послідовність символів між пробільними символами

- Прапор inword визначає, чи відноситься поточний символ до слова

если c не является пробельным символом и inword ложно
установить inword в истину и посчитать слово

if c является пробельным символом и inword истинно
установить флаг inword в ложь

Продовження коду

```
33     if (prev != '\n')
34         p_lines = 1;
35     printf("characters = %ld, words = %d, lines = %d, ",
36           n_chars, n_words, n_lines);
37     printf("partial lines = %d\n", p_lines);
38
39     return 0;
40 }
```


```
E:\GDisk\[College]\[фёютш яёюёрьётрээ ёр рыуюёшёь|ўэ| ьютш]\тхьр 02. -Ёшэўшыш ёётюёхээ ёр тшьюзрээ яёюёрь\Code\wordcnt.e
Enter text to be analyzed (| to terminate):
Reason is a
powerful servant but
an inadequate master.
|
characters = 55, words = 9, lines = 3, partial lines = 0
```



wordcnt.c

Умовна операція ?:

- Єдиний тернарний оператор у мові C.
 - Форма запису: Вираз1 ? Вираз2 : Вираз3;
 - Аналог оператора if-else

`if (y < 0) x = -y;`
`else x = y;`  `x = (y < 0) ? -y: y;`

```
1  /* paint.c -- uses conditional operator */
2  #include <stdio.h>
3  #define COVERAGE 350          // square feet per paint can
4  int main(void)
5  {
6      int sq_feet;
7      int cans;
8
9      printf("Enter number of square feet to be painted:\n");
10     while (scanf("%d", &sq_feet) == 1)
11     {
12         cans = sq_feet / COVERAGE;
13         cans += ((sq_feet % COVERAGE == 0)) ? 0 : 1;
14         printf("You need %d %s of paint.\n", cans,
15               cans == 1 ? "can" : "cans");
16         printf("Enter next value (q to quit):\n");
17     }
18
19     return 0;
20 }
```


```
E:\GDisk\College\Шкільний курс програмування\Ср руююЕшСь|ўэ| ьютш\ҫхьр 02. -Ешэўшыш ёётюёхээ Ср
Enter number of square feet to be painted:
349
You need 1 can of paint.
Enter next value (q to quit):
351
You need 2 cans of paint.
Enter next value (q to quit):
q
```




paint.c

Оператори безумовного переходу (jump statements)

- Допоміжні оператори для циклів
 - continue** пропускає решту ітерації і переходить до наступної
 - break** перериває цикл, в якому його було викликано, та переходить до наступної інструкції



```
while ( (ch = getchar() ) !=EOF)
{
    blahblah(ch);
    if (ch == '\n')
        continue;
    yakyak(ch);
}
blunder(n,m);
```



```
while ( (ch = getchar() ) !=EOF)
{
    blahblah(ch);
    if (ch == '\n')
        break;
    yakyak(ch);
}
blunder(n,m);
```

Використання оператора break для виходу з циклу

```
1  /* break.c -- uses break to exit a loop */
2  #include <stdio.h>
3  int main(void)
4  {
5      float length, width;
6
7      printf("Enter the length of the rectangle:\n");
8      while (scanf("%f", &length) == 1)
9      {
10         printf("Length = %0.2f:\n", length);
11         printf("Enter its width:\n");
12         if (scanf("%f", &width) != 1)
13             break;
14         printf("Width = %0.2f:\n", width);
15         printf("Area = %0.2f:\n", length * width);
16         printf("Enter the length of the rectangle:\n");
17     }
18     printf("Done.\n");
19
20     return 0;
21 }
```

```
E:\GDisk\College\{фёзютш яёюуёрьетрээ ёр рыуюёшёь|ўэ| ьютш}\тхър 02. 4ёшэўш
Enter the length of the rectangle:
4
Length = 4.00:
Enter its width:
5
Width = 5.00:
Area = 20.00:
Enter the length of the rectangle:
q
Done.
```



break.c

Використання оператора break в операторі галуження switch-case

- Оператор switch-case зручніший за if-else, коли кожен випадок галуження визначається одним значенням – *міткою (label)*.
 - Спочатку отримується значення умови
 - Потім переглядається список міток (тут – 1, 2, 3)
 - Якщо жодна з них не дорівнює значенню з умови, відбувається перехід до мітки default (якщо вона є) або перехід до наступної інструкції

```
switch(число)
{
    case 1: оператор 1;
           break;
    case 2: оператор 2;
           break;
    case 3: оператор 3;
           break;
    default: оператор 4;
}
statement 5;
```

```

1 #include <stdio.h>
2 #include <ctype.h>
3 int main(void)
4 {
5     char ch;
6     printf("Give me a letter of the alphabet, and I will give ");
7     printf("an animal name\nbeginning with that letter.\n");
8     printf("Please type in a letter; type # to end my act.\n");
9     while ((ch = getchar()) != '#')
10    {
11        if('\n' == ch)
12            continue;
13        if (islower(ch))    /* lowercase only */
14            switch (ch)
15            {
16                case 'a' :
17                    printf("argali, a wild sheep of Asia\n");
18                    break;
19                case 'b' :
20                    printf("babirusa, a wild pig of Malay\n");
21                    break;
22                case 'c' :
23                    printf("coati, racoonlike mammal\n");
24                    break;
25                case 'd' :
26                    printf("desman, aquatic, molelike critter\n");
27                    break;
28                default :
29                    printf("That's a stumper!\n");
30            }    /* end of switch */
31        else
32            printf("I recognize only lowercase letters.\n");
33        while (getchar() != '\n')
34            continue;    /* skip rest of input line */
35        printf("Please type another letter or a #.\n");
36    }    /* while loop end */
37    printf("Bye!\n");
38
39    return 0;
40 }

```

Демонстрація оператора switch-case

```

E:\GDisk\College\4еэютш яЁюуЁрьстрээ Ёр рыуюЁшЁь|ўэ| ьютш|ўхър 02.4Ёшэўшыш ёЁтюЁхээ Ёр тшьюэрээ яЁюуЁрь\Code\animals.exe
Give me a letter of the alphabet, and I will give an animal name
beginning with that letter.
Please type in a letter; type # to end my act.
a
argali, a wild sheep of Asia
Please type another letter or a #.
b
babirusa, a wild pig of Malay
Please type another letter or a #.
k
That's a stumper!
Please type another letter or a #.
#
Bye!

```

- Програма перевіряє тільки перший символ введеного тексту



animals.c

```

1 // vowels.c -- uses multiple labels
2 #include <stdio.h>
3 int main(void)
4 {
5     char ch;
6     int a_ct, e_ct, i_ct, o_ct, u_ct;
7
8     a_ct = e_ct = i_ct = o_ct = u_ct = 0;
9
10    printf("Enter some text; enter # to quit.\n");
11    while ((ch = getchar()) != '#')
12    {
13        switch (ch)
14        {
15            case 'a' :
16            case 'A' : a_ct++;
17                      break;
18            case 'e' :
19            case 'E' : e_ct++;
20                      break;
21            case 'i' :
22            case 'I' : i_ct++;
23                      break;
24            case 'o' :
25            case 'O' : o_ct++;
26                      break;
27            case 'u' :
28            case 'U' : u_ct++;
29                      break;
30            default : break;
31        }
32    }
33    printf("number of vowels:  A   E   I   O   U\n");
34    printf("                    %4d %4d %4d %4d %4d\n",
35           a_ct, e_ct, i_ct, o_ct, u_ct);
36
37    return 0;
38 }

```

Обробка багатьох міток

```

E:\GDisk\[College]\[фёэютш яёюуёрьётрээ ёр рыуюёшёь|ўэ| ьютш]\тхьр 02. 4ёшэўшяш ёёстюёхээ ё
Enter some text; enter # to quit.
This is TEXT
#
number of vowels:  A   E   I   O   U
                  0   1   2   0   0

```



vowels.c

- Оператор switch-case не можна використовувати для дробових значень умови та складно застосувати, якщо значення має належати діапазону.
 - Тоді краще використовувати if-else

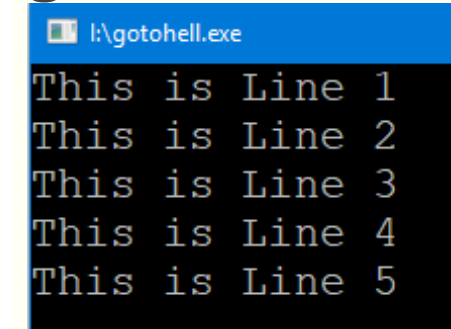
Оператор goto та проблема goto hell

Оператор goto

- Успадкований від таких мов, як BASIC та FORTRAN
- Складається з ключового слова goto та мітки.
- Потреби в цьому операторі немає, використовувався для простішого переписування коду з BASIC та FORTRAN
- Оператор є потенційним джерелом проблем

```
1  #include <stdio.h>
2
3  int main()
4  {
5      puts("This is Line 1");
6      goto this;
7  that:
8      puts("This is Line 3");
9      goto theother;
10 backhere:
11     puts("This is Line 5");
12     goto end;
13 this:
14     puts("This is Line 2");
15     goto that;
16 theother:
17     puts("This is Line 4");
18     goto backhere;
19 end:
20
21     return(0);
22 }
```

goto hell



```
I:\gotohell.exe
This is Line 1
This is Line 2
This is Line 3
This is Line 4
This is Line 5
```




ДЯКУЮ ЗА УВАГУ!

Наступне запитання: трансляція програмного коду