Лабораторна робота №19
Директиви препроцесора

Мета роботи: закріплення навичок роботи із директивами препроцесора.

Зміст:

Короткі теоретичні відомості

Директиви препроцесора

Директиви препроцесора – це інструкції препроцесору *C*, які виконуються **до** трансляції програми.

Директиви препроцесора дозволяють змінити текст програми, наприклад, замінити деякі лексеми в тексті, вставити текст з іншого файлу, заборонити трансляцію частини тексту і т.п.

Всі директиви препроцесора починаються зі знаку #.

Після директив препроцесора крапка з комою не ставляться.

В таблиці нижче наведені основні директиви препроцесора.

Директива	Опис
#define	Директива макропідстановки
#if	Управління умовною компіляцією
#elif	
#else	
#endif	
#error	Дозволяє вивести повідомлення про помилку та призупинити виконання компіляції
#ifdef	Визначають визначено ідентифікатор чи ні
#ifndef	
#line	Дає команду компілятору змінити збережений ним номер рядка та ім'я файлу на задані
#undef	Видалення поточного визначення ідентифікатора
#include	Додавання вмісту заданого файлу в інший файл
#pragma	Інструкція компілятору. Набір інструкцій специфічний для кожного конкретного виробника зазвичай описаний в керівництві до компілятора.

Директива макропідстановки #define

Директива макропідстановки #***define*** служить для текстово] заміни певного тексту програми деяким виразом.

Директива #define має такі синтаксичні форми:

#define ідентифікатор текст

#define ідентифікатор

#define ідентифікатор(список параметрів) текст

Перша синтаксична форма цієї директиви використовується для простої текстової заміни. В усіх місцях програми, де буде зустрічатися ***ідентифікатор,*** його буде замінено на на ***текст*** на етапі препроцесорної обробки програми .

Наприклад:

```
#define   E_VAL   2.71828   // в тих місцях де написано E_VAL в
                            // результаті препроцесорної обробки буде
                            // виконана заміна на 2.71828
```

В другій синтаксичній формі, де одразу за ключовим словом ***define*** знаходиться тільки ідентифікатор, визначається ідентифікатор без будь-якого значення. Визначені таким чином ідентифікатори можна використовувати в якості умови для директиви ***#ifdef***. В цьому випадку буде повернуто значення «істина».

Остання синтаксична форма директиви ***#define*** виконує макропідстановку із формальними параметрами, тобто створює так званий *макрос*. Ця ситаксична форма відрізняється від інших наявністю формальних параметрів – це величини, які знаходяться в круглих дужках одразу після ідентифікатора. Параметрів може бути декілька і вони повинні слідувати один за одним через кому. Формальні параметри в тексті макроозначення відзначають позиції, на які повинні бути підставлені фактичні аргументи макровиклику. Кожен формальний параметр може з'явитися в тексті макроозначення кілька разів.

Наприклад:

```
#define    MAX(a, b)     ( ((a) > (b)) ? (a) : (b) )
```

В наведеному прикладі створюється макрос із ім’ям *MAX* із двома формальними параметрами. Макрос визначає, який із параметрів має більше значення.

Нижче показано, як можна використати цей макрос:

```
int x = 10;
int y = 20;
int z = MAX(x, y);
printf("max(%d, %d) = %d\n", x,y,z);
```

Передостанній рядок прикладу дуже схожий на виклик функції, але механізм виклику функції та використання макросу принципово різний. У разі макросу, виконується просто текстова заміна, тобто ще до компіляції цей рядок буде замінено текстом із визначення макросу, а замість параметрів будуть підставлені їх значення.

Імена ідентифікаторів, які використовуються в директиві ***#define*** повинні відповідати усім правилам, які стосуються інших ідентифікаторів в мові *C*.

Текст-замісник може бути довільним і до нього не має певних вимог. Якщо текст-замісник зручно розмістити на декількох рядках, то для переносу тексту на новий рядок використовують символ зворотної похилої риски – «\».

Є певні обмеження використання макросів, наприклад, якщо імя макросу розташовано всередині рядка, який укладений в лапки, то текстова заміна виконана не буде.

Область видимості імені, визначеного за допомогою директиви препроцесора *#define*, починається від його визначенням і закінчується кінцем файлу.

При використанні директиви макропідстановки слід дотримуватися обережності. В директиві define використовується безпосередня текстова заміна, тому при недбалому програмуванні можуть виникнути помилки.

Наприклад:

```
#define square(x)  (x*x)
…
int a = 1;
int b = square(a+2);
```

В останньому прикладі визначено макрос, який обчислює квадрат формального параметра.

В останньому рядку прикладу ми очікуємо, що змінна *b* буде дорівнювати 9, але насправді *b* буде рівне 5. Дійсно, якщо виконати безпосередню текстову заміну, то отримаємо

```
int b = (a+2*a+2);
```

замість останнього рядка прикладу.

Щоб убезпечити себе від подібного типу помилок, краще брати усі параметри всередині тексту-заміни в круглі дужки. Для останнього прикладу коректний макрос повинен виглядати наступним чином:

```
#define square((x)*(x))
```

Також з обережністю слід використовувати формальні параметри, які інкрементуються чи декрементуються, оскільки такі значення можуть дати непередбачувані результати.

Директива #undef

Директива ***#undef*** використовується для скасування дії директиви ***#define***.

Усі ідентифікатори, об'явлені за допомогою директиви ***define*** буде видимі від моменту оголошення і до кінця файлу. Якщо потрібно скасувати визначення константи, то використовують директиву ***#undef***.

Якщо застосувати директиву ***#undef*** до ідентифікатора який не був визначений, то це не є помилкою.

Синтаксис цієї директиви:

#undef ідентифікатор

Директиви умовної компіляція

До директив умовної компіляції відносяться директиви . ***#if***, ***#elif***, ***#else***, та ***#endif***. Ці директиви дозволяють керувати самим ходом виконання препроцесорної обробки. Директиви умовної компіляції позволяють вибірково включати той чи інший текст програми в залежності від значення умови, що обчислюється під час компіляції.

Директива препроцесора ***#if*** працює аналогічно умовному оператору *if*, але значення виразів в умові повинні бути відомі на етапі компіляції програми, тобто умова повинна складатися із констант.

При виконанні цієї директиви спочатку обчислюється константний вираз, записаний в рядку одразу після директиви ***#if***:

- якщо значення константного виразу НЕ дорівнює нулю, то в текст програми будуть включені всі подальші рядки аж до ***#endif***, або ***#elif*** чи ***#else***,
- якщо значення після директиви ***#if*** дорівнює нулю і через декілька рядків зустрічається директива ***#else***, то в текст програми будуть включені всі рядки, які слідують одразу після директиви ***#else*** і до директиви ***#endif,***
- якщо значення після директиви ***#if*** дорівнює нулю і через декілька рядків зустрічається директива ***#elif***, то обчислюється вираз після директиви ***#elif*** і далі виконуються перевірки на рівність виразу нулю: якщо цей вираз має ненульове значення, то в текст програми будуть включені всі наступні рядки аж до ***#endif***, або ***#elif*** чи ***#else***.

Синтаксис директив умовної компіляції:

#if константний-вираз1
дії1
#elif константний-вираз2
дії2
#else
дії3
#endif

Інколи із директивами умовної компіляції зручно використовувати слово *defined*, коли потрібно знати – визначена та чи інша константа чи ні.

Наприклад:

```
#if defined(N)
printf(“You can create array.”);
#else
printf(“Error! N is not defined.”);
#endif
```

В цьому прикладі перевіряється визначення константи *N*, яка є довжиною масиву. Якщо *N* не визначено, то друкується повідомлення про помилку.

Директиви #ifdef та #ifndef

Досить часто в програмах потрібно значи – визначена та чи інша константа чи ні. Для цього зручно користуватися спеціальними директивами препроцесора ***#ifdef*** та ***#ifndef***. Ці директиви призначені для перевірки того, визначена чи ні константа із заданим ім'я.

З використанням цих директив попередній приклад можна переписати наступним чином:

```
#ifdef N
printf(“You can create array.”);
#endif

#ifndef N
printf(“Error! N is not defined.”);
#endif
```

Директива #error

Директива ***#error*** ПРИПИНЯЄ виконання компіляції проекту та видає ПОВІДОМЛЕННЯ у вікні помилок компілятора.

Попередній приклад можна переписати із використанням директиви ***#error***:

```
#ifndef N
#error Error! N is not defined
#endif
```

Принципова відмінність цього коду полягає в тому, що ми побачимо це повідомлення не у консолі програми під час її виконання, а у вікні із помилками на етапі компіляції.

Корисні посилання

Виробником компілятора можуть надаватися специфічні і нестандартні директиви. Наприклад, для компілятора фірми *Microsoft* більш детальну інформацію про директиви препроцесора можна отримати за посиланням:

https://docs.microsoft.com/cpp/preprocessor

Робоче завдання

Скласти програму, згідно варіанту.

Номер варіанта	Завдання 1
1	За допомогою макросу обчислити периметр квадрата. Написати програму для перевірки роботи макросу, підставивши в якості довжини сторони значення *a+1*, де *a* – змінна типу *int* із будь-яким значенням.
2	За допомогою макросу обчислити синус кута, якщо кут задається у градусах. Написати програму, в якій значення кута у градусах зчитується із клавіатури та пораховане значення синуса виводиться на екран.
3	За допомогою макросу надрукувати ASCII код символу, який передається в макрос. Написати програму, яка буде зчитувати із клавіатури будь-який символ і друкувати його ASCII на екран.
4	Створити макрос, який знаходить максимальне серед трьох значень. Написати програму для перевірки роботи макросу, підставивши в якості параметрів значення *a+1*, $b-2$ та $2c$, де *a*, *b* та *c* – змінні типу *int* із будь-якими значеннями.
5	За допомогою макросу обчислити площу квадрату. Написати програму для перевірки роботи макросу, підставивши в якості довжини сторони значення *a+10*, де *a* – змінна типу *int* із будь-яким значенням.
6	За допомогою макросу обчислити косинус кута, якщо кут задається у градусах. Написати програму, в якій значення кута у градусах зчитується із клавіатури та пораховане значення косинусу виводиться на екран.

7	Створити макрос для перетворення великих літер на малі. Створити програму, яка зчитує з клавіатури рядок (який повинен містити і великі, і малі літери), та перетворює його на рядок із малих літер із тим самим змістом.
8	За допомогою макросу обчислити периметр прямокутника. Написати програму для перевірки роботи макросу, підставивши в якості довжин сторін значення $a+1$ та $b-2$, де a та $b-$ змінні типу *int* із будь-якими значеннями.
9	За допомогою макросу обчислити тангенс кута, якщо кут задається у градусах. Написати програму, в якій значення кута у градусах зчитується із клавіатури та пораховане значення тангенсу виводиться на екран.
10	Створити макрос для перетворення малих літер на великі. Створити програму, яка зчитує з клавіатури рядок (який повинен містити і великі, і малі літери), та перетворює його на рядок із великих літер із тим самим змістом.
11	За допомогою макросу обчислити площу прямокутника. Написати програму для перевірки роботи макросу, підставивши в якості довжин сторін значення $a+7$ та $b-3$, де a та $b-$ змінні типу *int* із будь-якими значеннями.
12	За допомогою макросу обчислити арксинус, виводячи значення кута у градусах. Написати програму, в якій параметри, який буде передано в макрос, зчитується із клавіатури, а пораховане значення кута виводиться на екран у градусах.
13	Створити макрос, який знаходить мінімальне серед трьох значень. Написати програму для перевірки роботи макросу, підставивши в якості параметрів значення $a+1$, $b-2$ та $2c$, де a, b та c – змінні типу *int* із будь-якими значеннями.

14	За допомогою макросу обчислити периметр трикутника. Написати програму для перевірки роботи макросу, підставивши в якості довжин сторін значення $a+1$, $b-2$ та $2c$, де a, b та c – змінні типу *int* із будь-якими значеннями.
15	За допомогою макросу обчислити арккосинус, виводячи значення кута у градусах. Написати програму, в якій параметри, який буде передано в макрос, зчитується із клавіатури, а пораховане значення кута виводиться на екран у градусах.
16	Створити макрос який би отримував би певне ціле значення та друкував на екран: якщо це літера, то друкував на екран її ASCII код, в противному випадку – повідомлення «Error!». Створити програму, яка зчитує з клавіатури певний символ та друкує на екран його код чи повідомлення про помилку.
17	За допомогою макросу обчислити периметр п’ятикутника. Написати програму для перевірки роботи макросу, підставивши в якості довжин сторін значення $a+1$, $a-2$, $b-2$, $b+3$ та $3c$, де a, b та c – змінні типу *int* із будь-якими значеннями.
18	За допомогою макросу обчислити арктангенс, виводячи значення кута у градусах. Написати програму, в якій параметри, який буде передано в макрос, зчитується із клавіатури, а пораховане значення кута виводиться на екран у градусах.
19	Створити макрос який би зчитував с екрану число, та перетворював його в літеру відповідно до ASCII таблиці. Якщо код не літерою, то повинно бути надруковане повідомлення «Error!». Створити програму, для перевірки роботи цього макросу.
20	Створити макрос для перетворення великих літер на малі, а малих – на великі. Створити програму, яка зчитує з клавіатури рядок (який повинен містити і великі, і малі літери), та перетворює його на рядок із інверсією малих літер на великі та навпаки.

Контрольні питання

1. Що таке директива препроцесора?
2. В яких випадках зручно користуватися директивами препроцесора?
3. Які директиви препроцесора ви знаєте?
4. Директива препроцесора ***#error*** припиняє виконання компіляції чи ні?