

Міністерство освіти і науки України
Національний університет “Львівська Політехніка”

Лабораторна робота №6
З дисципліни
“Програмування частина 2”

Виконав:
Студент групи АП-11
Гишка Остап

Прийняв:
Чайковський І.Б.

Львів-2024

«Загальна структура програми на мові С»

Мета роботи: ознайомитися із загальною структурою побудови програм на мові С, навчитися використовувати функції введення та виведення даних.

Теоретичні відомості

Програма на мові С складається з однієї або більше функцій і хоча б одна з них повинна називатися `main()`. Опис функції складається з заголовку та тіла. Заголовок у свою чергу містить директиви препроцесора типу `#include` тощо, що під'єднують бібліотечні файли та специфікують перетворення тексту програми перед компіляцією; а також ім'я функції. Ознакою імені функції служать круглі дужки. Тіло функції поміщається в фігурні дужки та є набором операторів (команд), кожен із яких закінчується символом `“ ; “` - крапка з комою. Елементом програми є коментар - частина тексту програми для пояснення окремих операторів, що входять до її складу. Коментар не впливає на виконання операторів і записується таким чином : `//текст коментарю` або так: `/* текст коментарю*/` . В першому випадку коментар має бути єдиним у рядку або в кінці рядка. Другий спосіб дозволяє записувати коментар будь-де в тексті програми. Оголошення змінної задає ім'я та атрибути змінної. Визначення змінної, крім задання імені та її атрибутів, приводить до виділення для неї пам'яті. Програма може містити довільне число директив, вказівок компілятору, оголошень та визначень. Їх синтаксис розглядатиметься нижче. Порядок появи цих елементів у програмі є важливий

Формат	Тип інформації, що виводиться
%d	Десяткове ціле число
%c	Один символ
%s	Рядок символів
%e	Число з плаваючою точкою, експонентний запис
%f	Число з плаваючою точкою, десятковий запис
%g	Використовується замість записів %f або %e , якщо він коротший.
%u	Десяткове ціле число без знаку
%o	Вісімкове ціле число без знаку
%x	Шістнадцяткове ціле число без знаку

Модифікатор	Значення
—	Аргумент буде друкуватися з лівої позиції заданої ширини. Друк аргументів закінчується в правій крайній позиції поля. Приклад: %-10d
Рядок цифр	Задає мінімальну ширину поля. Більше поле буде використовуватися, якщо друковане число або рядок не вміщуються в початковому полі. Приклад: %4d
Рядок цифр	Визначає точність: для типів даних із плаваючою точкою - число друкованих цифр праворуч від десяткової точки; для символічних рядків максимальне число друкованих символів. Приклад: %4.21 (дві десяткові цифри для поля шириною в чотири символи)
L	Відповідний елемент даних має тип <code>long</code> , а не <code>int</code> . Приклад: %ld

Приклад 1

```
#include <stdio.h>
main()
{
    int z;
    int w;
    int x=1;
    int y=2;
    z=x+y;
    w=y-x;
    printf("%d",z);
    printf("%d",w);
}
```

3

1

Приклад 2

```
#define PI 3.14159
#include <stdio.h>
main()
{
    int a=5;
    float b=23.5;
    int c=31000;
    printf("%d матеріалів тканини коштувало %f гривень.\n",a,b);
    printf("Значення числа PI рівне %f.\n", PI);
    printf("IBM сумісні комп'ютери набули широкого розповсюдження. \n");
    printf("%c%d\n",'$', c);
}
```

5 метрів тканини коштувало 23.50 гривень

Значення числа PI 0.000000

IBM сумісні комп'ютери набули широкого розповсюдження.

&31000

Приклад 3

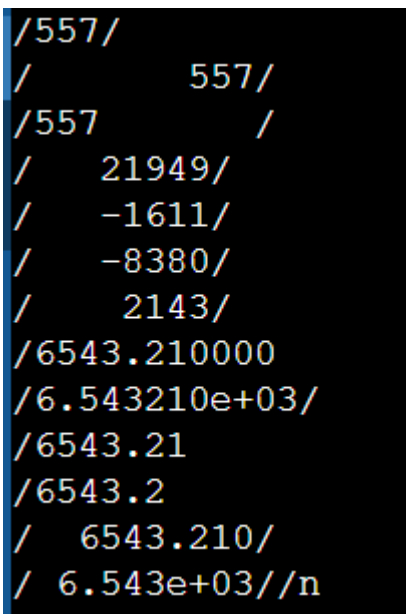
```
#include <stdio.h>
main()
{
    printf("/%d\n",557);
    printf("/%10d\n",557);
    printf("/%-10d\n",557);
}
main()
{
```

```

    printf("/%8d\n",21949);
    printf("/%8d\n",-1611);
    printf("/%8d\n",-8380);
    printf("/%8d\n",2143);
}

main()
{
    printf("/%f\n", 6543.21);
    printf("/%e\n", 6543.21);
    printf("/%4.2f\n", 6543.21);
    printf("/%3.1f\n", 6543.21);
    printf("/%10.3f\n", 6543.21);
    printf("/%10.3e\n", 6543.21);
}

```



```

/557/
/      557/
/557      /
/   21949/
/   -1611/
/   -8380/
/    2143/
/6543.210000
/6.543210e+03/
/6543.21
/6543.2
/  6543.210/
/ 6.543e+03//n

```

Приклад 4

```

#define riadok "Чудова погода"
main()
{
    printf("/%2s\n", riadok);
    printf("/%15.s\n", riadok);
}

main()
{
    printf("%d\n",557);
    printf("%o\n",557);
    printf("%x\n",557);
    printf("%d\n",-557);
}

```

```
/Чудова погода/  
/ /
```

```
557  
1055  
22d  
-557
```

Приклад 5

```
#include<stdio.h>
```

```
main()
```

```
{  
    int vik;  
    char name[30];  
    printf("Vash vik?\n");  
    scanf("%d",&vik);  
    printf("Vvedit vashe imya\n");  
    scanf("%s",name);  
    printf("Ptyvit %s Jakomy(iy) %d rokiv",name,vik);  
}
```

```
Vash vik?  
18  
Vvedit vashe imya  
Ostap  
Ptyvit Ostap Jakomy(iy) 18 rokiv
```

Приклад 6

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
#define STOP '*'
```

```
main()
```

```
{  
    char ch;  
    ch=getchar();  
ml : if(ch != STOP);  
    {  
        putchar(ch);  
        ch=getchar(); goto ml;} }
```

```
X  
X  
3  
3  
*  
*  
  
^J
```

Контрольні запитання

1) Структура програми на мові C:

Програма на мові C складається з функцій, які можуть бути визначені користувачем або використовуватися зі стандартних бібліотек. Зазвичай програма починається з функції `main()`, яка є вихідним пунктом виконання програми.

2) Ідеологія організації операцій введення-виведення в мові C:

У мові C операції введення-виведення відбуваються через стандартні бібліотечні функції, такі як `printf()` і `scanf()`, які визначені у бібліотеці `stdio.h`. Ці функції дозволяють здійснювати форматоване виведення і введення даних відповідно.

3) Синтаксис функцій `printf()` і `scanf()`:

```
printf("формат_строки", список_аргументів);  
scanf("формат_строки", &змінні);
```

4) Основні типи форматів при звертанні до функцій `printf()` і `scanf()`:

`%d` - для цілих чисел

`%f` - для дійсних чисел

`%c` - для символів

`%s` - для рядків

5) Модифікатори форматів при звертанні до функцій `printf()` і `scanf()`:

`%` - вказує на початок специфікатора формату

`*` - вказує на пропуск полів в аргументі

6) Відмінності при застосуванні функцій `printf()` і `scanf()`:

`printf()` використовується для виведення даних на екран у вказаному форматі.
`scanf()` використовується для отримання даних з клавіатури у вказаному форматі.

7) Застосування функцій `getchar()` і `putchar()`:

`getchar()` - отримує наступний символ зі стандартного вводу.

`putchar()` - виводить один символ на стандартний вивід.

8) Пояснення змісту і обґрунтування результатів виконаних прикладів:

Для пояснення змісту та обґрунтування результатів прикладів потрібно конкретизувати приклади, які були виконані. Вони можуть включати в себе використання функцій `printf()`, `scanf()`, `getchar()`, `putchar()` та інших функцій введення-виведення. Результати виконання прикладів будуть залежати від введених даних та самого коду програми.