Laboratorul 1. Unelte de programare

Compilarea (Java, Pascal, C, etc.)

Constă în traducerea propriu-zisă de către un compilator din limbajul de programare ales în cod mașină, care va fi executat direct pe calculator.

Interpretarea (Python, JavaScript, etc.)

Implică faptul că programul nu va fi rulat direct de calculator, ci de un alt program, interpretor, care este deja compilat în cod mașină.

Compilatorul GCC

- compilator multi-frontend și multi-backend, cu suport pentru limbaje precum C, C++, Java.
- se apelează din linia de comandă, folosind diferite opțiuni, în funcție de rezultatul care se dorește (ex: opțiuni de optimizare).

Utilizare GCC

Fie programul hello.c.

- pentru a compila programul vom folosi comanda:

```
gcc hello.c
```

- comanda anterioară va produce un executabil cu numele a.out ce poate fi rulat astfel:

```
./a.out
```

- putem da un alt nume executabilului (ex: hello) prin intermediul opțiunii -o:

```
gcc -o hello hello.c
```

Opțiuni GCC

Opțiune	Efect
-o nume_fișier	Numele executabilului rezultat va fi nume_fișier.
-I cale_către_fișiere_antet	Caută fișiere antet și în calea specificată.
-L cale_către_biblioteci	Caută fișiere bibliotecă și în calea specificată.
-l nume_bibliotecă	Link-editează biblioteca specificată. **Atenție!** Nume_bibliotecă poate diferi de numele fișierului asociat bibliotecii (ex. **pentru includerea bibliotecii de funcții matematice, fișierul antet este math.h, iar biblioteca este m**).
-W tip_warning	Afișează tipurile de advertismente specificate.
-c	Compilează și asamblează, dar nu link-editează.
-S	Se oprește după faza de compilare, fără să asambleze.

Exemplu utilizare GCC

```
gcc -o hello hello.c -lm -Wall
```

Compilează și link-editează fișierul hello.c, cu includerea bibliotecii matematice, afișând toate avertismentele. Fișierul de ieșire se va numi hello.

Utilitarul Make

- **determină automat care sunt fișierele modificate** și **declașeanză** comenzile necesare pentru **recompilarea lor**.
- necesită existența unui fișier de tip **Makefile** care descrie **relațiile de dependență** între diferitele fișiere din care se compune programul și care specifică **regulile de actualizare** pentru fiecare fișier în parte.

Citirea datelor

```
printf("format", expr1, ..., exprn);
```

Atenție! Este necesar ca pentru fiecare expresie să existe un specificator de format și viceversa.

Sintaxa unui descriptor de format

```
%[-][lung][.frac][h|l|L]descriptor
```

Câmp	Descriere
-	Aliniere la stânga.
lung	În absența lui lung, expresia va fi afișată cu atâtea caractere câte conține.
frac	Numărul de cifre după virgulă (precizia) cu care se face afișarea.
h	Marchează un short int.
l	Marchează un long int.
L	Marchează un long double.

Afișarea datelor

```
scanf("format", &var1, ..., &varn);
```

Atenție! Funcția scanf primește adresele variabilelor în care are loc citirea.

Sintaxa unui descriptor de format

```
%[*][lung][l]descriptor
```

Câmp	Descriere
*	Valoarea citită nu se atribuie unei variabile.
lung	Lungimea câmpului din care se face citirea.
l	Indică un long int sau un real double.

Descriptori

Descriptor	Descriere
d	Întreg cu semn în baza 10.
u	Întreg fără semn în baza 10.
o	Întreg fără semn în baza 8.
x sau X	Întreg fără semn în baza 16.
c	Caracter.
s	Șir de caractere.
f	Real zecimal de forma [-]xxx.yyyyyy (implicit 6 cifre după virgulă).
e sau E	Real zecimal în notație exponențială.
g	Analog cu e, E și f dar afișarea se face cu număr minim de cifre zecimale.