Operatori

matematici

+	-	*	/	%

relazionali

<	>	<=	>=	==	!=

logici

&&	||	!

Precedenza degli operatori

Operatore	Precedenza
chiamata a funzione	
! + - & (operatori unari)	
* / %	
+ -	
< <= > = >	
== !=	
&&	
||	
=	

Costrutti decisionali

if

if (condizione**)**
  istruzione$_V$;
[**else**
  istruzione$_T$;]
	if (condizione**) {**
	  istruzione$_{V1}$;
	  ...
	  istruzione$_{Vn}$;
	} [**else {**
	  istruzione$_{T1}$;
	  ...
	  istruzione$_{Tn}$;
	}]
		if (condizione$_1$**)**
		  istruzione$_{V1}$;
		else if (condizione$_2$**)**
		  istruzione$_{V2}$;
		...
		else if (condizione$_n$**)**
		  istruzione$_{Vn}$;
		else
		  istruzione$_e$;

switch

switch (espressione**) {**
  case valore$_1$**:** istruzioni$_1$;
  [**break;**]
  case valore$_2$**:** istruzioni$_2$;
  [**break;**]
  case valore$_3$**:** istruzioni$_3$;
  [**break;**]
  ...
  case valore$_n$**:** istruzioni$_n$;
  [**break;**]
  [**default:** istruzioni$_d$;]
}

Costrutti iterativi

while

while (condizione di ripetizione del ciclo**)** istruzione;	**while (**condizione di ripetizione del ciclo**){** istruzione$_1$; istruzione$_2$; ... istruzione$_n$; **}**

for

for (espressione di inizializzazione**;** condizione di ripetizione del ciclo**;** espressione di aggiornamento**)** istruzione;	**for (**espressione di inizializzazione**;** condizione di ripetizione del ciclo**;** espressione di aggiornamento**) {** istruzione$_1$; istruzione$_2$; ... istruzione$_n$; **}**

do-while

do istruzione; **while (**condizione di ripetizione del ciclo**);**	**do{** istruzione$_1$; istruzione$_2$; ... istruzione$_n$; **}while (**condizione di ripetizione del ciclo**);**

Direttive

include

#include nome_libreria

define

#define simbolo valore

Librerie ANSI

stdio.h

Prototipo	Scopo
FILE *fopen(char *filename, char *mode);	Apre un file **filename** in modalità **mode** e restituisce il puntatore al file. Restituisce NULL nel caso di errore.
void fclose(FILE *fp);	Chiude il file individuato da **fp**.

int getc(FILE *fp);	Legge un carattere dal file individuato da **fp** e passa al successivo.
int getchar(void);	Legge un carattere dallo standard input.
int putc(char ch, FILE *fp);	Scrive il carattere **ch** sul file individuato da **fp**.
int putchar(char ch);	Scrive il carattere **ch** sullo standard output.
char * gets(char buffer[]);	Legge una linea di testo da standard input e la memorizza in **buffer** .
int fgets(char buffer[], int max, FILE *fp);	Legge una linea di testo di al più **max** caratteri dal file individuato da **fp** e la memorizza in **buffer**. Il terminatore di riga viene memorizzato in buffer.
int fputs(char *buffer, FILE *fp);	Scrive la stringa **buffer** nel file individuato da **fp**.
void printf(char *buffer, ...);	Scrive la stringa **buffer** sullo standard ouput.
void fprintf(FILE *fp, char *buffer, ...);	Scrive la stringa **buffer** sul file individuato da **fp**.
int scanf(char *format, ...);	Acquisisce i dati dallo standard input secondo il formato **format** e li memorizza nelle variabili i cui indirizzi vengono indicati dopo la ,. Restituisce il numero di valori correttamente acquisiti.
int fscanf(FILE *fp, char *format, ...);	Acquisisce i dati dal file individuato da **fp** secondo il formato **format** e li memorizza nelle variabili i cui indirizzi vengono indicati dopo la ,. Restituisce il numero di valori correttamente acquisiti.

stdlib.h

Prototipo	**Scopo**
int abs(int n);	Restituisce il valore assoluto di **n**.
void *malloc(int nBytes);	Alloca un blocco di memoria di dimensioni **nByte** e restituisce il puntatore al primo indirizzo del blocco. Se non c'è sufficiente memoria restituisce NULL.
void free(void *p);	Libera la memoria associata al puntatore **p**, allocata in precedenza mediante una malloc.

math.h

Prototipo	**Scopo**
double ceil(double x)	Arrotonda **x** all'intero superiore rappresentato in virgola mobile.
double cos(double x)	Restituisce il coseno dell'angolo **x** (espresso in radianti).
double exp(double x)	Restituisce e^{x}.
double fabs(double x)	Restituisce il valore assoluto di **x** (tipo double).
double floor(double x)	Arrotonda **x** all'intero inferiore.
double fmod(double x, double y)	Restituisce il resto della divisione **x** / **y** rappresentato in virgola mobile.
double log(double x)	Restituisce il logaritmo naturale di **x**.
double log10(double x)	Restituisce il logaritmo base10 di **x**.
double pow(double x, double y)	Restituisce x^{y}.
double sin(double x)	Restituisce il seno dell'angolo **x** (espresso in radianti).
double sqrt(double x)	Resituisce la radice quadrata positiva di **x**.
double tan(double x)	Restituisce la tangente dell'angolo **x** (espresso in radianti).

ctype.h

Prototipo	**Scopo**

int isalpha(int ch);	Restituisce 0 se **ch** non è un carattere alfabetico, altrimenti un numero intero diverso da 0. ('A' - 'Z' oppure 'a' - 'z')
int isupper(int ch);	Restituisce 0 se **ch** non è un carattere maiuscolo, altrimenti un numero intero diverso da 0. ('A' - 'Z')
int islower(int ch);	Restituisce 0 se **ch** non è un carattere minuscolo, altrimenti un numero intero diverso da 0. ('a' - 'z')
int isdigit(int ch);	Restituisce 0 se **ch** non è una cifra, altrimenti un numero intero diverso da 0. ('0' - '9')
int isalnum(int ch);	Restituisce 0 se **ch** non è un carattere alfanumerico, altrimenti un numero intero diverso da 0. ('A' - 'Z' oppure 'a' - 'z' oppure '0' - '9')
int isspace(int ch);	Restituisce 0 se **ch** non è un carattere di spaziatura, altrimenti un numero intero diverso da 0. (spazio, tabulatore, a capo, nuova linea, tabulatore verticale, nuova pagina)
int toupper(int ch);	Se **ch** è un carattere alfabetico restituisce il corrispondente carattere maiuscolo, in caso negativo restituisce il carattere non modificato.
int tolower(int ch);	Se **ch** è un carattere alfabetico restituisce il corrispondente carattere minuscolo, in caso negativo restituisce il carattere non modificato.

string.h

Prototipo	Scopo
int strlen(char *s);	Restituisce la lunghezza della stringa **s**, escluso il terminatore '\0'
int strcmp(char *s1, char *s2);	Restituisce 0 se la stringa **s1** è uguale a **s2**, un numero negativo se **s1** viene prima di **s2** nell'ordine lessicografico, un numero positivo in caso contrario.
char *strcpy(char dest[], char *sorg);	Copia la stringa **sorg** nell'array **dest**, e restituisce ls stringa copiata.

Versione pdf stampabile