|  |  |
| --- | --- |
| **Nome e Cognome** |  |
| **Matricola** |  |
| **Corso**  1(AAAA - BARA) ☐ 2 (BARB – BOTS) ☐ 3 (BOTT – CAR) ☐ 4 (CAS – CORD) ☐ 5 (CORE – DIF) ☐ 6 (DIG – FIOR) ☐ 7 (FIOS - GIORD) ☐ 8 (GIORE – LANE) ☐ 9 (LANF – MARA) ☐ 10 (MARB – MOH) ☐ 11 (MOI – PAK) ☐ 12 (PAL – POLH) ☐ 13 (POLI – ROSA) ☐ 14 (ROSB – SIL) ☐ 15 (SIM – TR) ☐ 16 (TS – ZZ) ☐ E1 (AA – LZ) ☐ E2 (MA – ZZ) ☐ Poli@Home ☐ Es. (5 crediti)☐ | |

**Teoria**

**Domanda 1**

|  |  |
| --- | --- |
|  | *Risultato* |
| Dati i seguenti numeri nella rappresentazione in CA2 codificati in esadecimale  X: B2  Y: 2A  trasporli nella codifica binaria su 8 bit, mantenendo la rappresentazione in CA2, ed eseguirne la somma verificando l’eventuale presenza di OVERFLOW | Risultato somma:  Overflow (sì/no): |
| Passaggi | |

**Domanda 2**

|  |  |
| --- | --- |
| Sapendo che il parallelismo di una memoria è di 8 bit, dimensionare il BUS INDIRIZZI affinche’ si possa gestire un quantitativo di memoria pari a 32 KByte | Dimensione BUS INDIRIZZI (in bit): |
| Passaggi |  |

**Domanda 3**

|  |
| --- |
| Descrivere memoria centrale e memoria di massa indicandone le principali differenze. |
|  |

**Programmazione**

Si desidera scrivere un programma per analizzare il traffico del rientro dalle vacanze.

Il nome del file contenente le informazioni sul traffico delle vacanze è passato come primo argomento della linea di comando, ogni riga del file si riferisce ad una singola persona ed è formattata come segue:

<NOME> <COGNOME> <LOCALITA> <DISTANZA> <MEZZO>

dove NOME (stringa) e COGNOME (stringa) sono il nome e cognome di ciascuna persona, LOCALITA (stringa) è il nome della località dove tale persona ha trascorso le vacanze, DISTANZA (numero intero) è la distanza in km fra la località e l’abitazione della persona, MEZZO (stringa) è il mezzo utilizzato per il viaggio.

Si facciano inoltre le seguenti assunzioni:

* Tutti i campi sono separati tra loro da uno o più spazi;
* Le stringhe sono lunghe al più 40 caratteri e non contengono spazi al loro interno;
* Il numero di persone non è noto a priori (e di conseguenza nemmeno il numero di righe del file);
* Il formato e il contenuto del file sono corretti.

Un secondo file, dal nome prefissato “trasporti.txt”, contiene le informazioni sui mezzi di trasporto nel seguente formato:

<MEZZO> <VELOCITA>

dove MEZZO (stringa) è il nome del mezzo di trasporto e VELOCITA (numero intero) è la sua velocità media in km/h. I due campi sono separati da uno o più spazi e i mezzi sono al massimo 500. Si assuma che il formato di questo file sia corretto e che siano presenti (una e una sola volta) tutti i mezzi che compaiono nel file precedente.

Il programma deve creare un terzo file, il cui nome viene specificato come secondo argomento sulla linea di comando, le cui righe conterranno, per ogni persona presente nel file in ingresso: il nome e cognome della persona e la durata del viaggio di ritorno in ore (con due cifre dopo la virgola), ciascun campo separato da un singolo spazio.

## Esempio

Ipotizzando che il primo file passato come argomento sulla riga di comando si chiami “elenco.txt” e contenga:

Paolo Rossi Springfield 500 bicicletta  
Mario Bianchi Aspen 50 piedi

Marco Verdi Maui 3000 aereo

Ipotizzando che il file “trasporti.txt” contenga:

bicicletta 40

piedi 6

mongolfiera 60

aereo 500

## Esecuzione del programma:

C:\> **mioprog.exe elenco.txt risultato.txt**

Il file “risultato.txt” dovrà contenere:

Paolo Rossi 12.50

Mario Bianchi 8.33  
Marco Verdi 6.00

**#include <stdio.h>**

**FILE \*fopen(char \*filename, char \* mode**) – Apertura di un file (mode: “r” lettura – “w” scrittura – “a” append)

**FILE \*freopen(char \*filename, char \* mode, FILE \*file\_pointer**) - Riassegna un file puntatore ad un file diverso.

**int fclose(FILE \*file\_pointer)** - Chiude un file

**int feof(FILE \*file\_pointer) -** Controlla se e' stato incontrato un end-of-file in un file.

**int fflush(FILE \*file\_pointer) -** Svuota il buffer di un file.

**int getchar(void) -** Legge un carattere da "stdin" (tastiera)

**int fgetc(FILE \*file\_pointer) -** Prende un carattere da un file

**char \*gets(char \*buffer) -** Legge una riga da "stdin" (tastiera)

**char \*fgets(char \*string, int maxchar, FILE \*file\_pointer) -** Legge una riga da un file.

**int printf(char \*format \_string, …) -** Scrive output formattato su "stdout" (schermo)

**int fprintf(FILE \*file\_pointer, char \*format\_string, …) -** Scrive output formattato in un file.

**int sprintf(char \*string, char \*format\_string, …) -** Scrive output formattato su una stringa

**int fputc(int c, FILE \*file\_pointer) -** Scrive un carattere in un file

**int putchar(int c) -** Scrive un carattere su "stdout" (schermo)

**int puts(char \*string)** - Scrive una stringa su "stdout" (schermo)

**int fputs(char \*string, FILE \*file\_pointer)** - Scrive una stringa in un file.

**int scanf(char \*format\_string, args)** - Legge input formattato da "stdin" (tastiera)

**int fscanf(FILE \*file\_pointer, char \*format string, args) -** Legge input formattato da file

**int sscanf(char \*buffer, char \*format\_string, args) -** Legge input formattato da una stringa

**EOF** – end of file (costante a valore negativo)

**NULL** - puntatore nullo (valore 0)

**#include <stdlib.h>**

**double atof(char \*string)** - Converte una stringa in un valore in floating point.

**int atoi(char \*string)** - Converte una stringa in un valore integer.

**int atol(char \*string)** - Converte una stringa in un valore long integer.

**void exit(int val)** – Termina il programma, restituendo il valore ‘val’.

**EXIT\_FAILURE** - costante per segnalare terminazione senza successo del programma con exit(); valore diverso da zero

**EXIT\_SUCCESS** - segnala terminazione con successo del programma con exit(); vale 0

**#include <string.h>**

**char \*stpcpy (char \*dest, char \*src) -** Copia una stringa in un'altra. Restituisce dest

**char \*strncpy(char \*s1, char \*s2, size\_t n) -** Copia i primi "n" caratteri di s2 in s1. Restituisce s1

**int strcmp(char \*s1, char \*s2) -** Confronta s1 e s2 per determinare l'ordine alfabetico (<0, s1 prima di s2, 0 uguali, >0 s1 dopo s2)

**int strncmp(char \*s1, char \*s2, size\_t n) -** Confronta i primi "n" caratteri di due stringhe.

**char \*strcpy(char \*s1, char \*s2) -** Copia s2 in s1. Restituisce s1

**int strlen(char \*string) -** Determina la lunghezza di una stringa.

**char \*strcat(char \*s1, char \*s2, size\_t n) -** Aggiunge s2 a s1. Ritorna s1

**char \*strncat(char \*s1, char \*s2, size\_t n) -** Aggiunge "n" caratteri di s2 a s1. Ritorna s1

**char \*strchr(char \*string, int c) -** Cerca la prima occorrenza del carattere ‘c’ in string; restituisce un puntatore alla prima occorrenza di c in s, NULL se non presente

**char \*strrchr(char \*string, int c) -** Cerca l'ultima occorrenza del carattere ‘c’ in string

**char\* strstr(char\* s, char\* t)** - Restituisce un puntatore alla prima occorrenza di t all'interno di s. Restituisce NULL se t non è presente in s.

**char\* strtok(char\* s, const char\* t) -** scompone s in token, i caratteri che delimitano i token sono contenuti in t. Restituisce il puntatore al token (NULL se non ne trova nessuno). Alla prima chiamata in s va inserita la stringa da scomporre e in t i caratteri che delimitano i vari token. Per operare sulla stessa stringa, alle successive chiamate al posto di s si deve passare NULL

**#include <ctype.h>**

**int isalnum(int c)** - Vero se ‘c’ e' alfanumerico.

**int isalpha(int c)** - Vero se ‘c’ e' una lettera dell'alfabeto.

**int iscntrl(int c)** - Vero se ‘c’ e' un carattere di controllo.

**int isdigit(int c)** - Vero se ‘c’ e' un numero decimale.

**int islower(int c)** - Vero se ‘c’ e' una lettera minuscola.

**int isprint(int c)** - Vero se ‘c’ e' un carattere stampabile.

**int ispunct (int c)** - Vero se ‘c’ e' un carattere di punteggiatura.

**int isspace(int c)** - Vero se ‘c’ e' un carattere spazio.

**int isupper(int c)** - Vero se ‘c’ e' una lettera maiuscola.

**int tolower(int c)** – Ritorna ‘c’ in minuscolo.

**int toupper(int c)** - Ritorna ‘c’ in maiuscolo.

**#include <math.h>**

**int abs (int n) –** valore assoluto intero

**long labs(long n) –** valore assoluto long

**double fabs (double x ) –** valore assoluto di x

**double acos(double x) -** arcocoseno

**double asin(double x) -** arcoseno

**double atan(double x) -** arcotangente

**double atan2(double y, double x) –** arcotangente di y/x.

**double ceil(double x) –** intero superiore a x

**double floor(double x) –** intero inferiore a x.

**double cos(double x) –** x in radianti

**double sin(double x) –** x in radianti

**double tan(double x) –** x in radianti

**double cosh(double x) –** coseno iperbolico

**double sinh(double x) –** seno iperbolico

**double tanh(double x) –** tangente iperbolica

**double exp(double x) -** ex

**double log(double x) -** log(x).

**double log10 (double x ) –** logaritmo base 10

**double pow (double x, double y) -** xy

**int rand (void) –** intero casuale tra 0 e RND\_MAX.

**int random(int max\_num) –** valore casuale tra 0 e max\_num.

**void srand(unsigned seed) –** inizializza la sequenza di valori casuali

**double sqrt(double x) –** radice quadrata

**#include <limits.h>**

**INT\_MAX -** Indica il più grande valore che è possibile rappresentare con un int.

**INT\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un int.

**LONG\_MAX -** Indica il più grande valore che è possibile rappresentare con un long.

**LONG\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un long.

**#include <float.h>**

**FLT\_MAX, DBL\_MAX -** Indica il più grande valore che è possibile rappresentare con un float (o double)

**FLT\_MIN, DBL\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un float (o double)