



Istituto Istruzione Superiore "G. Vallauri"

Settore Tecnologico: Elettronica ed Elettrotecnica - Informatica e Telecomunicazioni - Meccanica Meccatronica ed Energia

Settore Economico: Amministrazione Finanza Marketing - Turismo

Settore Professionale: Servizi Commerciali

Settore Liceale: Liceo Scientifico opzione Scienze Applicate

Fossano

Blocco C



Cairone Fiorentino
Giacchello Giuseppe
5^B Informatica
Anno Scolastico 2015/16
Tesina Esame di Stato

SOMMARIO

PREFAZIONE	4
INTRODUZIONE	4
PERCHÉ ABBIAMO SCELTO QUESTO PROGETTO?	4
TECNOLOGIE UTILIZZATE	5
LINGUAGGIO C	5
LINGUAGGIO C#	5
COMPILATORE GCC	5
FILE XML	6
STRUTTURA DEL SOFTWARE E DIFFICOLTÀ AFFRONTATE	7
DISPENSE BLOCCO C.....	13
<u>FORM PRINCIPALE</u>	<u>13</u>
<u>CREAZIONE DI VARIABILI E VETTORI</u>	<u>14</u>
<u>AGGIUNTA DI UN BLOCCO ALL'INTERNO DEI PANNELLI</u>	<u>15</u>
<u>ELIMINAZIONE DI UN BLOCCO ALL'INTERNO DEI PANNELLI</u>	<u>15</u>
<u>CREAZIONE NUOVO PROGETTO</u>	<u>16</u>
<u>SALVATAGGIO DI UN PROGETTO</u>	<u>16</u>
<u>APERTURA DI UN PROGETTO</u>	<u>16</u>
<u>ESECUZIONE DEL PROGRAMMA</u>	<u>17</u>
<u>DESCRIZIONE DEI BLOCCHI E LORO UTILIZZO</u>	<u>19</u>
VETTORI E VARIABILI.....	19
Casella a discesa per le variabili.....	19
Assegnazione di una variabile	19
Modifica di una variabile.....	20
Assegnazione/modifica elementi di un vettore.....	21
Calcolo lunghezza di un vettore	22
Assegnazione/modifica variabile inserendo un elemento di un vettore	23
OPERAZIONI.....	23
Generazione di numeri casuali (random).....	23
Radice quadrata di un numero.....	24
Potenza di un numero	25
CONTROLLI	26
Casella di testo.....	26
Casella di carattere	26
Casella dei numeri interi.....	26
Casella dei numeri reali (double).....	26
Casella a discesa per i valori booleani.....	27
Controllo if	27
Controllo if else	28
Stampa a video di un testo/variabile.....	29
Input di una variabile.....	30
CICLI	31
Ciclo precondizionale (while)	31
Ciclo postcondizionale (do while).....	32
Ciclo con il contatore (for)	32
<u>STAMPA DI UN PROGETTO (FILE ANSI C)</u>	<u>33</u>
<u>ESEMPI DI PROGETTI BLOCCO C</u>	<u>34</u>
GIOCO DELL'ALTO BASSO.....	34
SOMMA DEGLI ELEMENTI DI UN VETTORE	36

RICERCA DI ELEMENTO ALL'INTERNO DI UN VETTORE CON ELEMENTI DISORDINATI E RIPETUTI.....	37
ORDINAMENTO DEGLI ELEMENTI DI UN VETTORE DI CARATTERI (METODO PER SELEZIONE)	39
CONVERTITORE DI NUMERI DA DECIMALE A BINARIO	42

Prefazione

Introduzione

Questo manuale è una guida pratica per l'utilizzo di Blocco C, un software per la creazione di semplici applicazioni in console usando le istruzioni base del linguaggio ANSI (*American National Standard Institute*) C. Ogni istruzione è tradotta in un blocco colorato con l'obiettivo di semplificare l'apprendimento agli utenti alle prime armi (esempio studenti di prima e/o seconda superiore).

Scopo principale dell'applicazione è proprio quello di convertire i blocchi inseriti dal programmatore in istruzioni ANSI C.

Inoltre è possibile eseguire il progetto che, richiamando automaticamente un compilatore ANSI C integrato (GCC), manderà in esecuzione il programma realizzato dall'utente.

Oltre al salvataggio del progetto e la conversione in C, Blocco C è in grado di stampare e salvare in PDF il file sorgente *.c.

Perché abbiamo scelto questo progetto?

La scelta è stata effettuata per vari motivi. Il primo perché il C è un linguaggio ***general purpose***, cioè permette di risolvere varie tipologie di problemi. Il secondo per eliminare un suo grande svantaggio: a causa delle sue strutture più “elementari” rispetto agli altri linguaggi il suo utilizzo è più complesso se non si è un programmatore esperto. Grazie al blocco C lo abbiamo voluto rendere più accessibile agli utenti non esperti.

Tecnologie utilizzate

Linguaggio C

Il C, nato negli anni '70 e standardizzato nel 1989, è un linguaggio di programmazione ad alto livello di tipo imperativo poiché comunica esattamente al microprocessore cosa deve essere eseguito e determina un'esecuzione sequenziale delle istruzioni tranne nei casi delle strutture di controllo dove si effettuano dei salti. Da esso ne derivano altri come il C++ e il C#.

Linguaggio C#

Il C# è un linguaggio *object oriented*, sviluppato da Microsoft e utilizza il NET Framework. Una delle IDE (*Integrated Development Enviroment*), proprietaria Microsoft, che implementa C# è Visual Studio.

Le principali differenze con il C sono:

- In molte operazioni aritmetiche vengono controllati eventuali overflow.
- Gli oggetti dinamici non vengono deallocati tramite codice ma la loro rimozione viene gestita automaticamente dal **garbage-collector** quando non esistono più riferimenti a tali oggetti.
- È possibile ereditare da una sola classe.
- Le sole conversioni implicite consentite sono quelle che non espongono al rischio di perdita di dati causata dalla diversa tipologia di dato. Ad esempio non puoi convertire un char in integer in modo implicito.
- È possibile creare dei programmi visuali.

Compilatore GCC

Un compilatore è un programma traduttore che, preso in input un file contenente un programma sorgente scritto in un linguaggio ad alto livello, produce in output un file contenente un programma equivalente scritto in linguaggio macchina.

Il compilatore GCC permette di tradurre il file sorgente *.c in eseguibile.

Il processo di compilazione comprende:

- **Analisi lessicale:** i caratteri contenuti nel file sono suddivisi in parti elementari, che permettono di individuare operatori e identificatori. Vengono eliminati anche i caratteri superflui come spazi e commenti.
- **Analisi sintattica:** le parti elementari sono raggruppati in comandi e vengono verificati se ci sono errori di tipo sintattico come un operatore mancante all'interno di un'operazione.
- **Analisi semantica:** si verifica se le variabili sono utilizzate correttamente e se le espressioni sono corrette.
- **Generazione del codice intermedio:** è generato un codice più adatto per arrivare al codice macchina.
- **Selezione di istruzioni:** in questa fase viene generato il codice *assembly* corrispondente al codice sorgente. Dopo varie operazioni viene creato il codice macchina.
- **Ottimizzazioni:** viene ottimizzato il codice macchina per migliorare le prestazioni.
- **Allocazione dei registri:** limita il numero di registri utilizzati e li riserva per il programma.
- **Generazione del codice:** il codice macchina viene salvato nel file *.obj.

File XML

XML (*eXtensible Markup Language*) è un formato di rappresentazione di dati che racchiude ogni dato all'interno di Tag creando così una struttura ad albero. Viene utilizzato soprattutto per memorizzare database di piccola dimensione, trasportare dati da un ambiente all'altro e memorizzare informazioni di configurazione. Uno svantaggio è la scarsa economicità in termini di spazio di occupazione rispetto ad altri formati come il **CSV**.

Un albero XML è formato da un'intestazione: `<?xml version="1.0" encoding="UTF-8" ?>` e una `<root>...</root>` fissa, dove al suo interno sono presenti vari elementi figli.

La struttura è di tipo gerarchica. Ogni Tag XML può contenere attributi, come i Tag HTML.

Struttura del software e difficoltà affrontate

Siamo partiti dalla creazione dei vari blocchi colorati definendo degli **User Control personalizzati** per ognuno di esso. Ogni classe contiene metodi e attributi differenti per ogni tipologia di istruzione, per esempio il blocco del ciclo For contiene metodi che permettono l'aggiunta di altri blocchi e il ridimensionamento automatico. Invece, il blocco per l'assegnazione di una variabile non contiene questi metodi ma gestisce il caricamento delle variabili nelle varie ComboBox.

Il secondo step è stato gestire il **drag&drop** dei blocchi, quindi l'inserimento dinamico di quest'ultimi nel pannello *Main* e di conseguenza nei pannelli dei *cicli* e delle *condizioni*.

Successivamente abbiamo affrontato una delle fasi più importanti, **nonché la più complessa**, del nostro progetto, cioè la conversione di ogni istruzione visuale nel corrispondente codice testuale.

Parte della funzione per la traduzione dei blocchi:

```
private void traduciBlocchi(string tipo, Panel pnl)
{
    if (tipo == "main")
    {
        //scorro le variabili
        for (int i = 0; i < varGlobali1.pnl.Controls.Count; i++)
        {
            switch (varGlobali1.pnl.Controls[i].GetType().ToString())
            {
                case "Esame.variabile":
                    variabile v = (variabile)varGlobali1.pnl.Controls[i];
                    if (v.tipo != "STRING")
                    {
                        if (v.tipo == "BOOL")
                        {
                            if (!header.Contains("typedef enum {FALSE, TRUE} bool;"))
                                header += "typedef enum {FALSE, TRUE} bool; /*tipo booleano*/";
                            codice += "\n" + v.tipo.ToLower() + " " + v.var + " = FALSE;";
                        }
                        else
                        {
                            if (v.tipo == "DOUBLE")
                                codice += "\n" + v.tipo.ToLower() + " " + v.var + " = 0.0000;";
                            else
                            {
                                if (v.tipo == "INT")
                                    codice += "\n" + v.tipo.ToLower() + " " + v.var + " = 0;";
                                else
                                    codice += "\n" + v.tipo.ToLower() + " " + v.var + " = ' '";
                                codiceXML += "<var>" + v.tipo.ToUpper() + "$" + v.var + "</var>";
                            }
                        }
                    }
                    else
                    {
                        codice += "\nchar " + v.var + "[1000] = \"\"; /*tipo stringa in ANSI C*/";
                        codiceXML += "<var>" + v.tipo.ToUpper() + "$" + v.var + "</var>";
                    }
                }
                break;
                case "Esame.vettore":
                    vettore vet = (vettore)varGlobali1.pnl.Controls[i];
                    if (vet.tipo == "BOOL")
                    {
                        if (!header.Contains("typedef enum {FALSE, TRUE} bool;"))
                            header += "typedef enum {FALSE, TRUE} bool; /*tipo booleano*/";
                    }
                    codice += "\n" + vet.tipo.ToLower() + " " + vet.var + "[" + vet.lunghezza + "];";
                    codiceXML += "<vet>" + vet.tipo.ToUpper() + "$" + vet.var + "$" + vet.lunghezza + "</vet>";

                break;
            }
        }
    }
}
```

```
    }  
    }  
    }  
    for (int i = 0; i < pnl.Controls.Count; i++)  
    {  
        try  
        {  
            switch (pnl.Controls[i].GetType().ToString())  
            {  
                case "Esame.visualizzaTesto":  
  
                    visualizzaTestoTraduci((Panel)pnl.Controls[i].Controls["pnlPadre"].Controls  
                        ["pnlTesto"], ref codice);  
  
                    break;  
                case "Esame.InputBox":  
                    codiceXML += "<InputBox>";  
  
                    visualizzaTestoTraduci((Panel)pnl.Controls[i].Controls["visualizzaTesto1"].  
                        Controls["pnlPadre"].Controls["pnlTesto"], ref codice);  
  
                    ComboBox cmb = (ComboBox)pnl.Controls[i].Controls["cmbVar"];  
                    string v = cmb.Text;  
                    if (v != "")  
                    {  
                        if (varGlobali.lstVarDouble.Contains(v))  
                            codice += "\nscanf(\"%lf\", &" + v + ");";  
                        else  
                        if (varGlobali.lstVarInt.Contains(v))  
                            codice += "\nscanf(\"%d\", &" + v + ");";  
                        else  
                        if (varGlobali.lstVarChar.Contains(v))  
                            codice += "\nscanf(\"%c\", &" + v + ");";  
                        else  
                        if (varGlobali.lstVarString.Contains(v))  
                            codice += "\nscanf(\"%s\", " + v + ");";  
                    }  
                    else  
                        codice += "\nscanf(\"\", &" + v + ");";  
                    codice += "\ngetchar();";  
                    codiceXML += "<input>" + v + "</input>";  
                    codiceXML += "</InputBox>";  
                    break;  
                case "Esame.assegnaVariabile":  
break;  
                //CONTINUA  
            case "Esame.controlloIf":  
                controlloIf contIf = (controlloIf)pnl.Controls[i];  
                bloccoControllo bloccoCont = (bloccoControllo)contIf.Controls["bloccoControllo1"];  
                codiceXML += "<if>";  
                codice += "\n";  
                controlloIf(bloccoCont, "if");  
                codiceXML += "<figli>";  
                codice += "\n{";  
                traduciBlocchi("controlloIf", (Panel)contIf.Controls["panelIf"]);  
                codice += "\n}";  
                codiceXML += "</figli></if>";  
break;  
                //CONTINUA  
            }  
        }  
        catch (Exception ex)  
        {  
  
            errore += pnl.Controls[i].GetType().ToString().Substring(6) + ": " + ex.Message +  
                "\n";  
        }  
    }
```



```
    }  
}  
  
private void controlloIf(bloccoControllo bloccoCont, string condizione)  
{  
    codiceXML += "<bloccoCondizione>";  
    int qta = bloccoCont.Controls["pnlPadre"].Controls.Count - 1;  
    codice += condizione + "(";  
    CheckBox not = (CheckBox)bloccoCont.Controls["pnlPadre"].Controls["chkNot"];  
    if (not.Checked)  
    {  
        codice += "!(";  
        codiceXML += "!$";  
    }  
    for (int y = 1; y <= qta; y++)  
    {  
  
        casellaAndOr casellaAndor =  
            (casellaAndOr)bloccoCont.Controls["pnlPadre"].Controls["casellaAndOr" + y];  
  
        casellaConfronto casellaConfr =  
            (casellaConfronto)casellaAndor.Controls["casellaConfronto1"];  
        string testol = "";  
  
        if (casellaConfr.Controls["pnlPadre"].Controls["pnl1"].Controls[0].GetType().ToString()  
            == "Esame.casellaTesto")  
            testol = "\"\" +  
                casellaConfr.Controls["pnlPadre"].Controls["pnl1"].Controls[0].Controls[0].Text  
                +  
                "\"\"";  
        else  
            if (casellaConfr.Controls["pnlPadre"].Controls["pnl1"].Controls[0].GetType().ToString() ==  
                "Esame.casellaCarattere")  
                testol = "'" +  
                    casellaConfr.Controls["pnlPadre"].Controls["pnl1"].Controls[0].Controls[0].Text  
                    +  
                    "'";  
        else  
            testol = casellaConfr.Controls["pnlPadre"].Controls["pnl1"].Controls[0].Controls[0].Text;  
        string testo2 = "";  
        if (casellaConfr.Controls["pnlPadre"].Controls["pnl2"].Controls[0].GetType().ToString()  
            == "Esame.casellaTesto")  
            testo2 = "\"\" + casellaConfr.Controls["pnlPadre"].Controls["pnl2"].Controls[0].Controls[0].  
                Text + "\"\"";  
        else  
            if (casellaConfr.Controls["pnlPadre"].Controls["pnl2"].Controls[0].GetType().ToString() ==  
                "Esame.casellaCarattere")  
                testo2 = "'" + casellaConfr.Controls["pnlPadre"].Controls["pnl2"].Controls[0].Controls[0].  
                    ext + "'";  
        else  
            testo2 =  
                casellaConfr.Controls["pnlPadre"].Controls["pnl2"].Controls[0].Controls[0].Text;  
        codice += "(" + testol + " " +  
            casellaConfr.Controls["pnlPadre"].Controls["cmbConfronti"].Text + " " + testo2  
            + ")";  
  
        ComboBox cm = (ComboBox)casellaConfr.Controls["pnlPadre"].Controls["cmbConfronti"];  
        codiceXML +=  
            casellaConfr.Controls["pnlPadre"].Controls["pnl1"].Controls[0].GetType().To  
                String() + "_" + testol + "$" + cm.SelectedIndex.ToString() + "$" +  
            casellaConfr.Controls["pnlPadre"].Controls["pnl2"].Controls[0].GetType().To  
                String() + "_" + testo2;  
        if (casellaAndor.Controls["cmbAndOr"].Text != "")  
        {  
            codice += casellaAndor.Controls["cmbAndOr"].Text == "AND" ? "&&" : "||";  
            codiceXML += "$" + casellaAndor.Controls["cmbAndOr"].Text + "$";  
        }  
    }  
    if (not.Checked)  
        codice += ")";  
}
```

```
    codice += ")";  
    codiceXML += "</bloccoCondizione>";  
}
```

La funzione `traduciBlocchi` vuole due parametri: il tipo simbolico del pannello (“main”, “cicloFor”, “controlloIf”, ...) e il puntatore fisico al pannello contenente i blocchi da convertire. È composta principalmente da una condizione che verifica se il tipo del pannello è “main”. Se soddisfa il controllo, la funzione scorre il contenitore delle variabili, tramite un `for`, e le scrive nella stringa `codice`.

Fatto questo, in ogni caso, esegue un altro ciclo `for` che questa volta andrà ad analizzare il pannello che si è passato come parametro. Ogni blocco è identificato da un suo tipo (esempio: “Esame.controlloIf”) grazie al quale lo `switch` può identificarlo per poi poterci accedere e prelevare i dati, inseriti dall’utente, e comporre così il codice C e XML per il salvataggio. Nel caso dei cicli e dei controlli il metodo viene richiamato in modo ricorsivo passandogli come parametro tipo e puntatore al suo pannello interno.

La funzione `controlloIf` permette di leggere e convertire le condizioni delle `if`, `ifelse` e dei cicli `while` e `doWhile`.

Per quanto riguarda il salvataggio di ogni progetto, vengono salvati tre file in una cartella: un file `*.c`, un file `*.bloccoc` e un file `*.exe`.

- Il file `*.bloccoc` contiene la conversione dei blocchi in formato XML, che servirà per il salvataggio del programma. Infatti **solamente** questo file potrà essere aperto tramite Blocco C, per continuare o visualizzare un progetto già incominciato.
- Il file `*.c` contiene il codice testuale indentato, in ANSI C dei blocchi convertiti nella fase precedente. È offerto all’utente come file di output per il confronto e l’apprendimento, ma non potrà essere aperto direttamente in Blocco C.

Ogni file ha una struttura comune:

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <math.h>  
int main()  
{  
    srand(time(NULL) + getpid());/*gestisce eventuali numeri random*/  
    //codice dei blocchicovertiti in C  
    printf("Premi un pulsante per uscire...");  
    getchar();  
    return 0;  
}
```

- Il file `*.exe` contiene l’eleggibile del programma. Può essere avviato quando e dove si vuole.

Un altro aspetto importante è quello dell’esecuzione del programma creato dall’utente tramite blocco C. Ogni qualvolta si converte, se la cartella del progetto è esistente, automaticamente vengono aggiornati i tre file sopra citati. Dopo la conversione, il programma parte con l’esecuzione, se non ci sono stati problemi sintattici o logici, grazie al compilatore GCC contenuto nella cartella `bin/Debug` del progetto d’esame.

Codice della conversione dei blocchi e esecuzione:

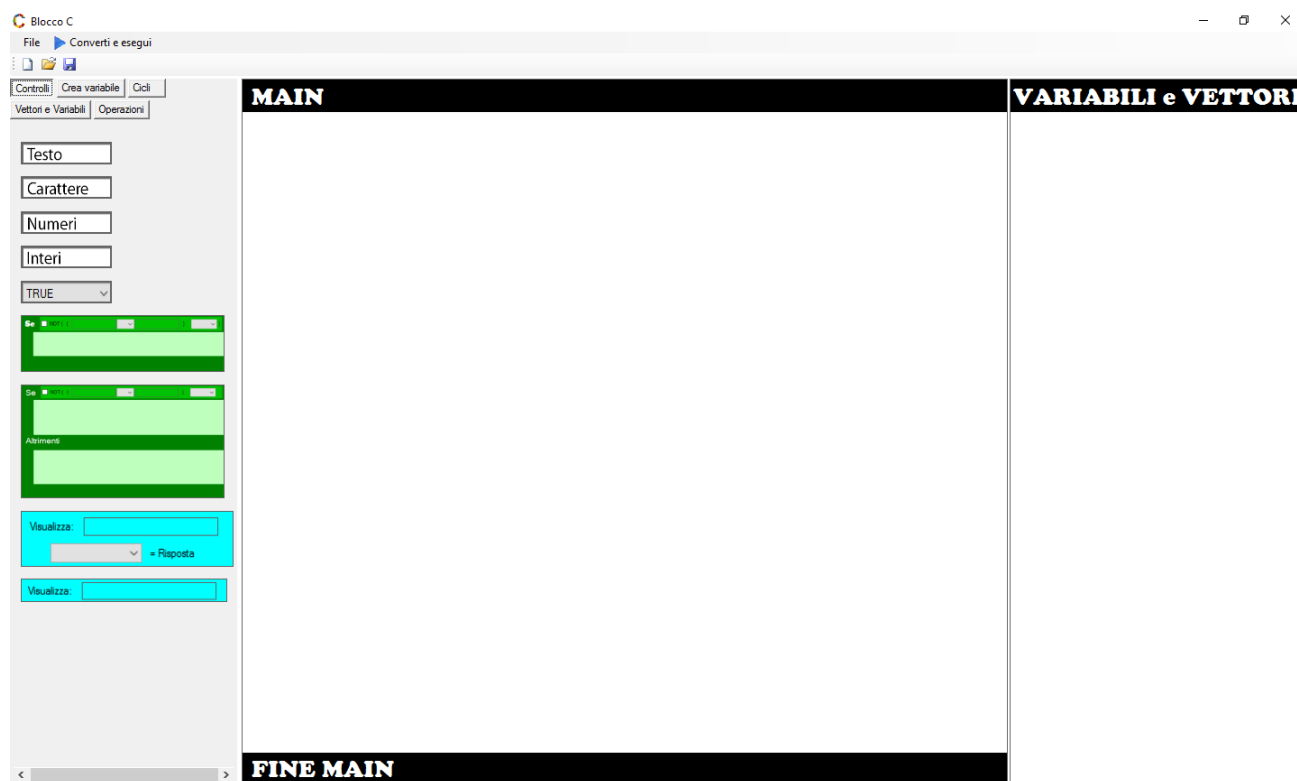
```
string nomeCartella = "";  
string fileC = "", fileEXE = "", fileXML = "";  
Process processoEXE;  
private void convertiToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
try
{
    if (File.Exists("test.exe"))
        File.Delete("test.exe");
    errore = "Errori:\n";
    header = "#include <stdio.h>\n#include <stdlib.h>\n#include <string.h>\n#include
        <math.h>\n";
    headerXML = "";
    codiceXML = "<root>";
    codice = "int main()\n{\n    srand(time(NULL) + getpid());/*gestisce eventuali numeri
        random*/\n";
    traduciBlocchi("main", pnlMain.pnl);
    codiceXML += "</root>";
    codice += "\nprintf(\"Premi un pulsante per uscire...\");\ngetchar();\nreturn 0;\n";
    if (errore != "Errori:\n")
        MessageBox.Show(errore, "ERRORI SUI BLOCCHI (OGGETTI MANCANTI)", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
}
else
{
    codiceFinale = header + "\n" + codice + "\n" + funzioni + "\n\n";
    codiceFinaleXML = headerXML + codiceXML;
    indentazioneFile();
    string programma = "", argomenti = "";
    if (nomeCartella == "" || !Directory.Exists(nomeCartella))
    {
        SaveFileDialog dlg = new SaveFileDialog();
        dlg.Filter = "Progetti Blocco C|*.bloccoc";
        dlg.DefaultExt = ".bloccoc";
        dlg.AddExtension = true;
        dlg.InitialDirectory =
            Environment.GetFolderPath(Environment.SpecialFolder.DesktopDirectory);
        DialogResult result = dlg.ShowDialog();
        if (result == DialogResult.OK)
        {
            fileC = dlg.FileName.Substring(dlg.FileName.LastIndexOf('\\'), (dlg.FileName.Length
                - 8) - dlg.FileName.LastIndexOf('\\')) + ".c";
            fileXML = dlg.FileName.Substring(dlg.FileName.LastIndexOf('\\') + 1);
            fileEXE = dlg.FileName.Substring(dlg.FileName.LastIndexOf('\\'),
                (dlg.FileName.Length - 8) - dlg.FileName.LastIndexOf('\\')) + ".exe";
            Directory.SetCurrentDirectory(dlg.FileName.Substring(0,
                dlg.FileName.LastIndexOf('\\')));
            nomeCartella = Directory.GetCurrentDirectory() + "" +
                dlg.FileName.Substring(dlg.FileName.LastIndexOf('\\'),
                dlg.FileName.LastIndexOf('.')
                - dlg.FileName.LastIndexOf('\\'));
            Directory.CreateDirectory(nomeCartella);
            Directory.SetCurrentDirectory(nomeCartella);
            fileC = nomeCartella + "\\\" + fileC;
            fileEXE = nomeCartella + "\\\" + fileEXE;
            fileXML = nomeCartella + "\\\" + fileXML;
            this.Text = "Blocco C - " + fileXML;
        }
    }
    else
    {
        nomeCartella = "";
        fileC = "";
        fileEXE = "";
        fileXML = "";
    }
}
if (fileC != "" && fileEXE != "")
{
    File.Delete(fileC);
    File.Delete(fileEXE);
    StreamWriter sw = new StreamWriter(fileC, false);
    sw.Write(codiceFinale);
    sw.Flush();
}
```

```
sw.Close();
programma = "cd " + Application.StartupPath + "\\bins\\gcc\n";
argomenti = String.Format("gcc.exe \"{0}\" -o \"{1}\"> \"{2}\" \n exit", fileC,
    fileEXE, Application.StartupPath + "\\errori.txt");
StreamWriter scriviBat = new StreamWriter(Application.StartupPath + "\\esegui.bat",
    false);
scriviBat.Write(programma + argomenti);
scriviBat.Flush();
scriviBat.Close();
Process.Start(Application.StartupPath + "\\esegui.bat");
Directory.SetCurrentDirectory(Application.StartupPath);
System.Threading.Thread.Sleep(1500);
StreamWriter swXml = new StreamWriter(fileXML, false);
swXml.Write(codiceFinaleXML);
swXml.Flush();
swXml.Close();
if (File.Exists(fileEXE))
{
    processoEXE = Process.Start(fileEXE);
}
else
{
    StreamReader sr = new StreamReader(Application.StartupPath + "\\errori.txt");
    string err = "ERRORI:\n";
    while (!sr.EndOfStream)
        err += sr.ReadLine() + "\n";
    sr.Close();
    MessageBox.Show(err, "ERRORI SUL FILE C ANSI, RICONTROLLARE I BLOCCHI",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
}
catch { }
}
```

DISPENSE BLOCCO C

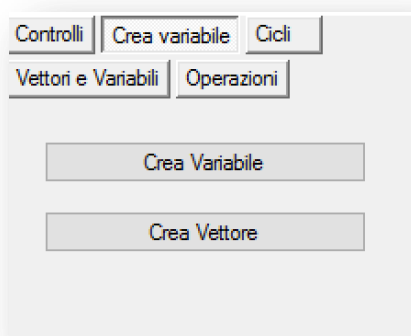
Form principale



Il Main di Blocco C è formato da:

- **Due pannelli:** uno per l'inserimento dei blocchi, tramite Drag&Drop, che staranno nel metodo `main()` del file ANSI C. L'altro, di sola lettura, che contiene i vettori e le variabili create.
- **Un menu:** dove è possibile salvare, aprire un file *.bloccoc o stampare il file PDF del file ANSI C.
- **Un bottone “Converti e esegui”:** converte i blocchi nel linguaggio ANSI C. La prima volta che si clicca dall'avvio del programma, viene chiesto di salvare il progetto dell'utente all'interno del computer. Fino a che non si farà “nuovo progetto”, contenuto nel menu, **Blocco C farà riferimento alla cartella salvata**, e le volte successive al click del bottone avvierà il codice scritto dall'utente aggiornandolo automaticamente.
- **Un menu dei blocchi:** dove l'utente, scelta la categoria, potrà trascinare un rettangolo sul pannello del Main e aggiungere così un blocco.

Creazione di variabili e vettori



Per creare una variabile o vettore bisogna andare nella sezione “Crea variabile”. Dopo di che si deve cliccare su uno dei due bottoni e in base a cosa si è scelto verranno visualizzati questi due Form:

Ogni variabile deve avere un nome diverso. Se per caso si dovesse sbagliare c'è un controllo automatico che avvisa l'utente. I tipi di ogni variabile sono: INT, CHAR, STRING (**vettore di Char**), BOOL (`'typedef enum {FALSE, TRUE} bool;'`: codice che consente di aggiungere il tipo booleano. Questo perché non esiste in ANSI C, e quindi viene creato dal programma scrivendo nel file in output questa riga fuori dal metodo principale `main()`) e DOUBLE. Mentre i vettori non hanno il tipo STRING. In aggiunta nei vettori bisogna anche inserire la lunghezza che deve essere un numero maggiore di 0. Le variabili verranno visualizzate nel seguente pannello. Il codice in ANSI C corrispondente per le variabili è:

```
tipo_variabile nome_variabile = valore_standard;  
esempio:  
int i = 0;
```

Per i vettori: `tipo_vettore nome_vettore[lunghezza_vettore];`

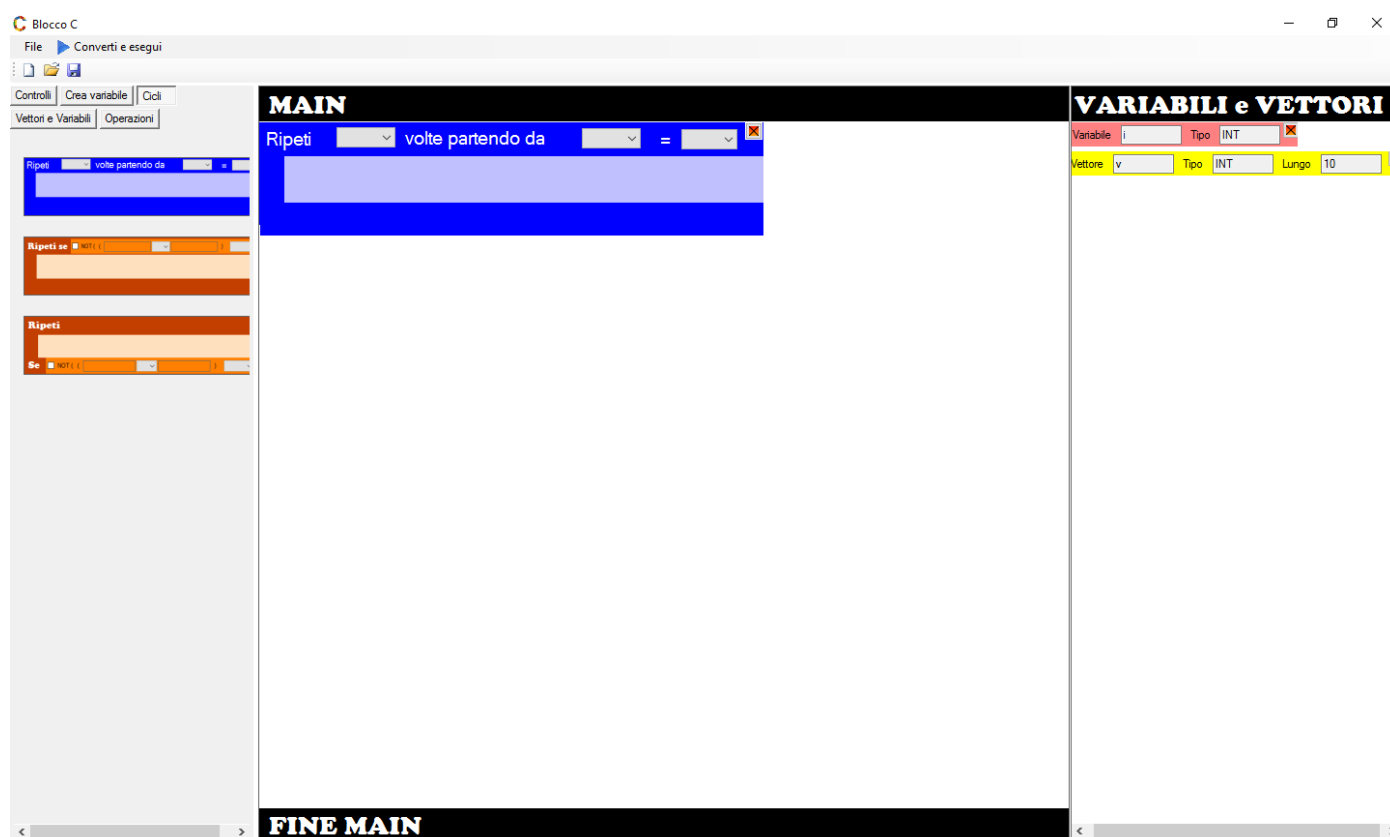
esempio:
int v[10];

VARIABILI e VETTORI					
Variable	<input type="text" value="i"/>	Tipo	<input type="text" value="INT"/>		
Vettore	<input type="text" value="v"/>	Tipo	<input type="text" value="INT"/>	Lungo <input type="text" value="10"/>	

È possibile eliminare una variabile creata cliccando sulla X in alto a destra.

Aggiunta di un blocco all'interno dei pannelli

Per aggiungere un blocco all'interno di un pannello (esempio il Main) bisogna trascinare al suo interno un blocco preso dal menu laterale. A questo punto verrà creato e aggiunto in coda a tutti gli altri. Un esempio:



Eliminazione di un blocco all'interno dei pannelli

Per eliminare un blocco all'interno dei pannelli, basta cliccare sulla X posta su ognuno di essi in alto a destra.



Creazione nuovo progetto

Per creare un nuovo progetto basta cliccare sul simbolo del file nuovo nel menu orizzontale posto in alto a sinistra, o scegliere la voce “Nuovo” da sottomenu “File”.



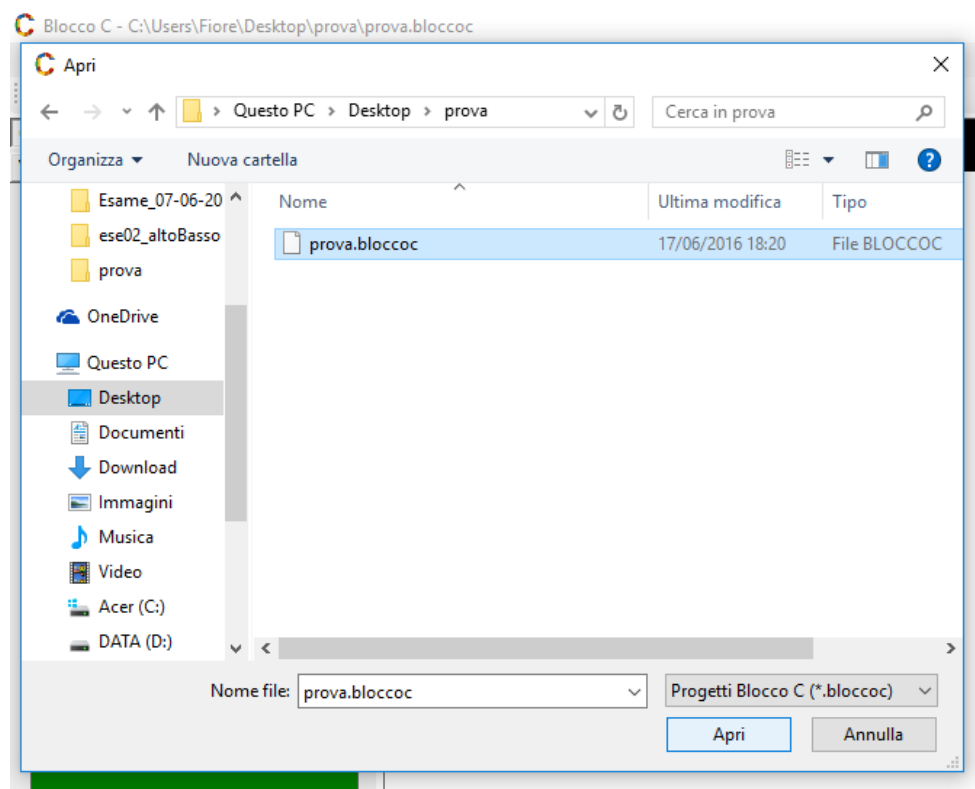
Salvataggio di un progetto

Per salvare il programma ci sono tre modi:

1. La prima volta che si clicca su “Converti e esegui” e non si è salvato il progetto viene chiesto all’utente di salvare il progetto con nome.
2. Se si clicca sul bottone salva, posto nel menu orizzontale, tramite il simbolino floppy disk o nel sottomenu “File->Salva”.
3. Nel sottomenu “File->salva con nome”. In questo caso, però crea solo una copia nel percorso specificato, ma non si farà riferimento a quest’ultimo come progetto aperto.

Apertura di un progetto

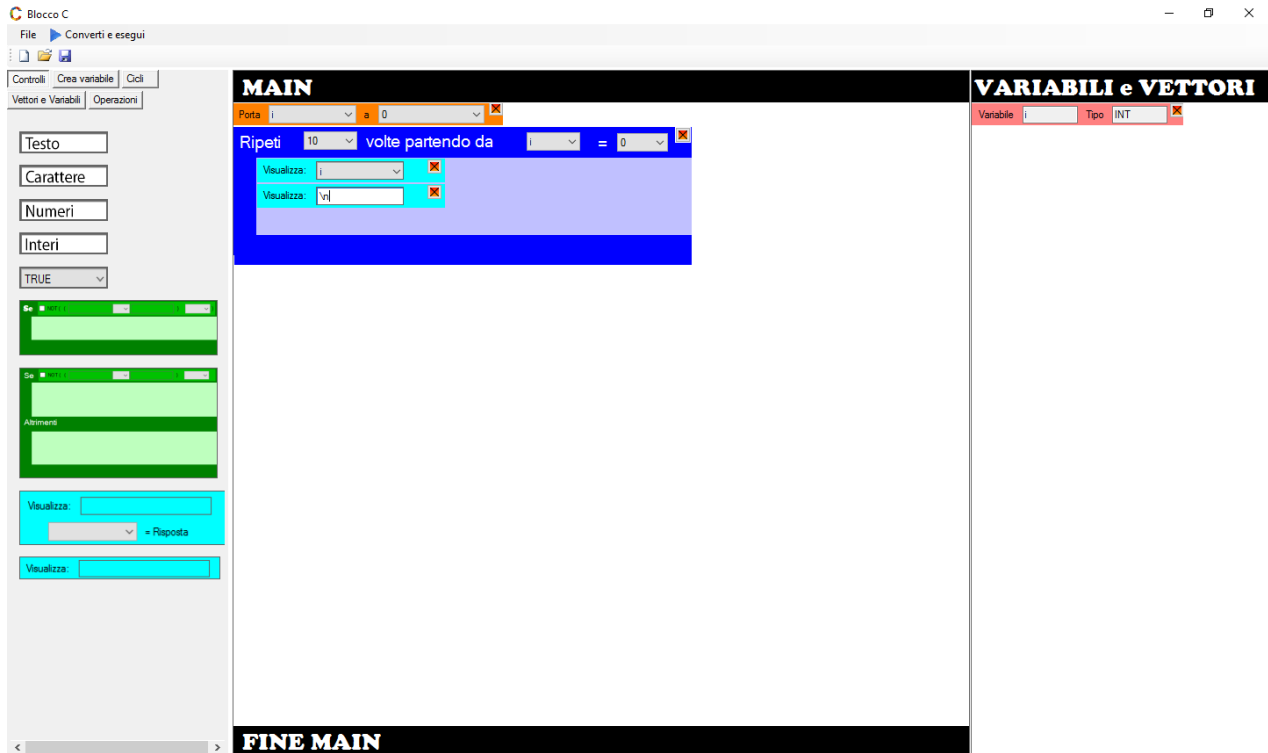
Dopo aver cliccato sul simbolo della cartella nel menu orizzontale posto in alto a sinistra, o scegliere la voce “Apri” dal sottomenu “File”, comparirà una finestra e si dovrà scegliere un file con estensione *.bloccoc.



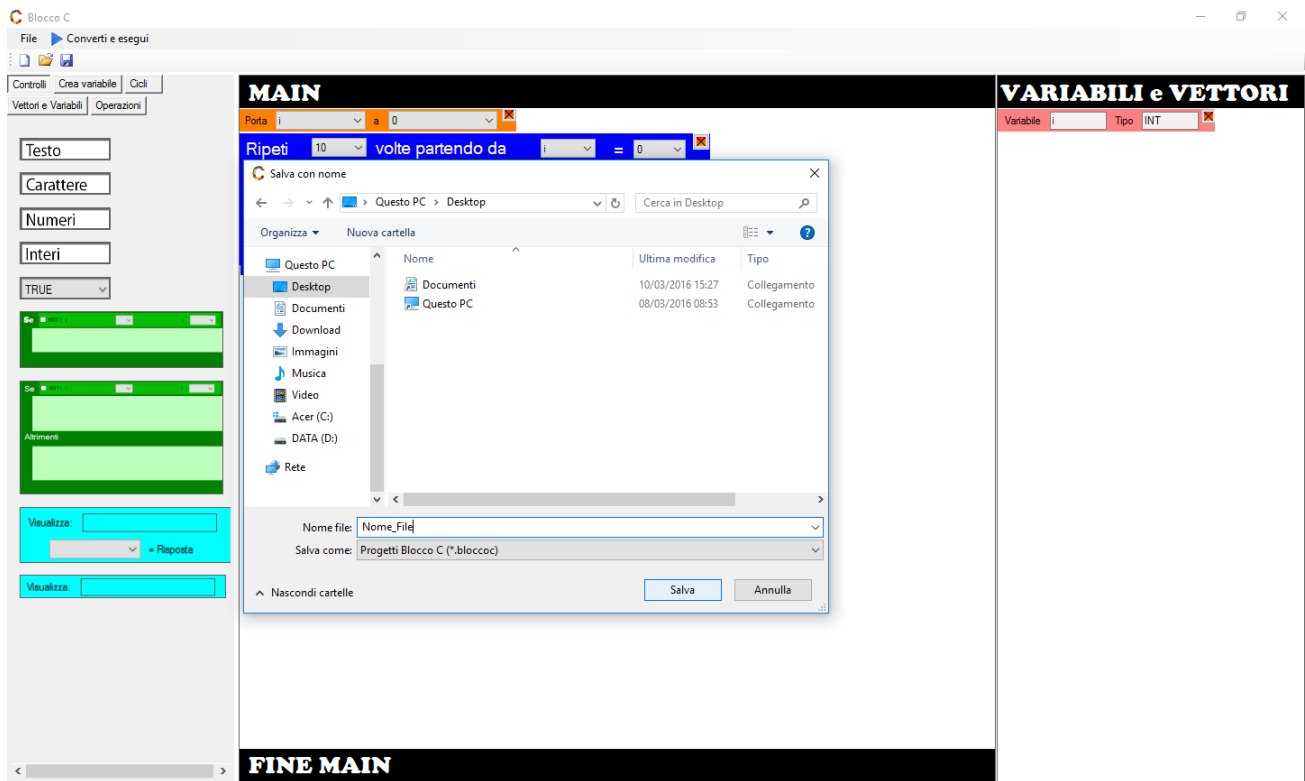
A questo punto il programma verrà aperto, visualizzando il suo contenuto nel pannello Main, e il suo nome nel titolo del Form principale.

Esecuzione del programma

Dopo aver “scritto” il codice, per avviare il programma bisogna cliccare sul bottone “Converti e esegui”.



Il programma verifica se è stato già salvato il progetto. Se non è così allora viene mostrata una finestra che permette di scegliere il percorso di salvataggio.



Dopo di che verrà eseguita la conversione automatica e se non ci sono errori il compilatore GCC creerà il file eseguibile, il file in ANSI C e il file XML (*.bloccoc) contenuti in una cartella. Infine verrà aperto in automatico il terminale dove verrà eseguito il programma.



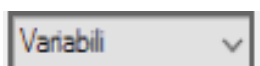
Ogni volta che si clicca sul bottone “Converti e esegui”, adesso, verrà in automatico aggiornato il progetto nella cartella scelta precedentemente.

Descrizione dei blocchi e loro utilizzo

Vettori e Variabili

I seguenti blocchi si trovano tutti nella sezione “Vettori e variabili” del menu a sinistra e consentono la gestione delle variabili e dei vettori.

Casella a discesa per le variabili



La seguente casella a discesa (Combobox), contiene tutte le variabili semplici (non i vettori) create dall’utente tramite il menu. Non è possibile inserirli direttamente all’interno del Main, come blocchi a sé stanti, ma vengono utilizzati all’interno di altri blocchi per diversi scopi che verranno citati più avanti.

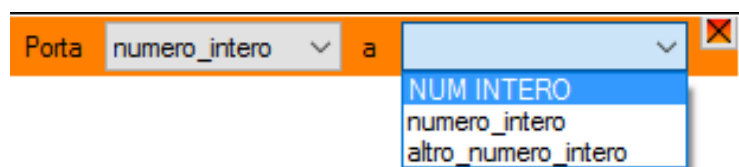
Assegnazione di una variabile



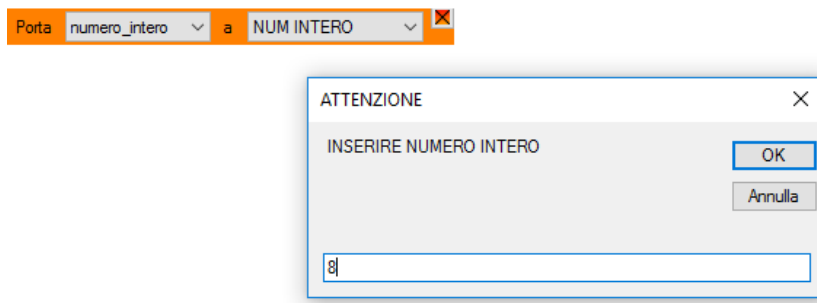
Il seguente blocco consente di assegnare ad una variabile un valore. È formato da due caselle a discesa: la prima permette di scegliere tra le varie variabili che sono state create dall’utente, la seconda permette di scegliere il valore da assegnare alla singola variabile. Il valore può essere: o inserito dall’utente scegliendo il primo elemento della seconda casella e immettendolo nell’apposita Form di input che compare; oppure, scegliendo dal secondo elemento in avanti, una variabile dello stesso tipo.

Esempio:

Si sceglie una variabili di tipo INT (numero intero)



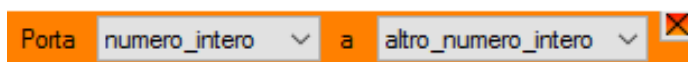
Se viene selezionato il primo elemento della seconda Combobox



Si immette il numero (in questo caso) e si preme ok. Il testo comparirà sulla Combobox, se non ci sono stati errori di input



Altrimenti si può selezionare un'altra variabile di tipo INT (in questo caso)



Questa cosa si può fare per qualsiasi tipologia di variabile, dato che il programma riconosce automaticamente il suo tipo. Esempio se si sceglie una variabile di tipo CHAR, si potrà immettere un carattere all'interno dell'Inputbox, e scegliere solo variabili di tipo carattere.

Nel caso di STRING o CHAR, i valori immessi dall'utente tramite Inputbox, si riconoscono perché racchiusi tra "" (doppi apici) per le stringhe o ' ' (singoli apici) per i caratteri.

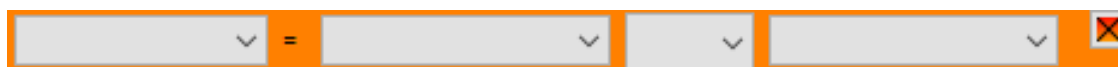
Il codice corrispondente in ANSI C è:

```
nome_variabile = valore/variabile;
```

esempio:

```
numero_intero = 8;  
numero_intero = altro_numero_intero;
```

Modifica di una variabile

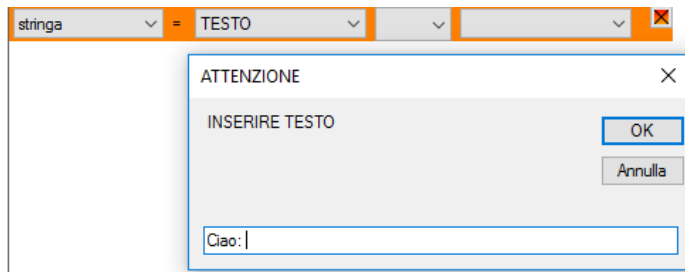
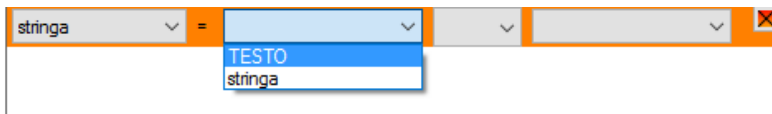


Il seguente blocco permette di modificare una variabile. È composto da 4 caselle a discesa:

1. La prima contiene tutte le variabili di tipo STRING, DOUBLE, INT (quelle che possiamo modificare attraverso un operatore). L'utente deve scegliere quella che vuole modificare.
2. La seconda contiene il primo operatore della operazione. In base a cosa si è scelto, è possibile inserire un valore da casella di input, scegliendo il primo elemento, o scegliere una variabile della stessa tipologia di quella da modificare.
3. La terza contiene gli operatori possibili da utilizzare per le operazioni. Nel caso dei DOUBLE e INT sono: +, -, *, /, [%] (resto della divisione solo per variabili INT). In caso di STRING l'operatore possibile è solo il +.
4. La quarta contiene il secondo operatore ed è gestito come la seconda Combobox.

Esempio:

Si sceglie una variabile di tipo STRING (testo). Il primo operatore è un testo



Invece il secondo, si sceglie un'altra variabile di tipo STRING, dopo aver selezionato l'operatore +



Il codice corrispondente in ANSI C è:

```
nome_variabile = valore/variabile (operatore) valore/variabile;
```

esempio:

```
stringa = "Ciao: " + nome;
```

È possibile utilizzare solo due operatori alla volta. Quindi, si consiglia, per le operazioni più complicate, di spezzarle in più blocchi e creare e utilizzare variabili di appoggio se necessario.

Assegnazione/modifica elementi di un vettore



Il seguente blocco permette di assegnare o modificare il valore di un elemento di un vettore. È composto da tre caselle a discesa: la prima contiene i vettori creati dall'utente, la seconda permette di inserire un valore numerico intero (posizione dell'elemento del vettore) tramite una casella di input (primo elemento della Combobox selezionata) oppure una variabile intera, la terza permette di inserire un valore dello stesso tipo degli elementi del vettore (primo elemento della Combobox selezionata) o una variabile dello stesso tipo.

Esempio:

Si sceglie un vettore di DOUBLE (numero reale). Si sceglie una posizione (in questo caso un numero: 10.3) e poi si sceglie un valore che viene inserito tramite Inputbox. I valori possono essere inseriti con o senza il carattere ',' (virgola) ma **NON con il carattere '.' punto** (la conversione verrà fatta in automatico dal programma).

Vettore double [0] = NUM REALE

ATTENZIONE

INSERIRE NUMERO REALE

OK

Annulla

10,3

Vettore double [0] = 10,3

Il codice corrispondente in ANSI C è:

```
nome_vettore[numero_intero/variabile_intera] = valore/variabile;
```

esempio:

```
double[0] = 10.3;  
double[i] = numero_double;
```

Calcolo lunghezza di un vettore

= Lunghezza ()

Il seguente blocco consente di richiamare una funzione, insieme di istruzioni richiamabili più volte all'interno del programma, che calcola la lunghezza di elementi di un vettore (numero di elementi che può contenere). È formato da due caselle a discesa: la prima contiene variabili intere create dall'utente, la seconda l'elenco dei vettori.

Esempio:

Si sceglie una variabile INT che conterrà il numero di elementi del vettore, e un vettore di tipo BOOL.

intero = Lunghezza (booleani)

Il codice corrispondente in ANSI C è:

```
#define lunghezzaVet(array) (sizeof(array)/sizeof(array[0])) //calcola la lunghezza del  
vettore  
int main(){
```

```
...  
nome_variabile_intera = lunghezzaVet(nome_vettore);  
...  
}
```

esempio:

```
#define lunghezzaVet(array) (sizeof(array)/sizeof(array[0])) //calcola la lunghezza del  
vettore  
  
int main() {  
    ...  
    intero = lunghezzaVet(booleani);  
    ...  
}
```

Dove viene definita una funzione `lunghezzaVet(array)` che è inserita nel file ANSI C la prima volta che viene aggiunto un blocco di questo tipo.

Assegnazione/modifica variabile inserendo un elemento di un vettore

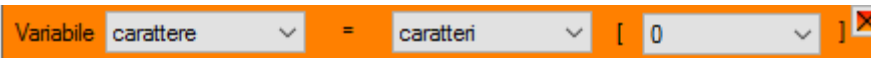


The screenshot shows a code generation interface with an orange background. It contains three dropdown menus: the first is labeled 'Variable' and has a downward arrow; the second is followed by an equals sign; the third is followed by an opening square bracket, a dropdown menu, and a closing square bracket. A red 'X' icon is at the end of the row.

Il seguente blocco consente di assegnare o modificare una variabile, inserendo un elemento contenuto in un vettore. È formato da tre caselle a discesa: la prima consente di selezionare una variabile di qualsiasi tipo, la seconda consente di scegliere un vettore dello stesso tipo della variabile, la terza consente di scegliere l'indice dell'elemento del vettore (numero intero o variabile intera).

Esempio:

Si sceglie una variabile CHAR (carattere) e il primo elemento di un vettore CHAR



The screenshot shows the same code generation interface as before, but with specific values selected in the dropdown menus: 'carattere' in the first, 'caratteri' in the second, and '0' in the third. A red 'X' icon is at the end of the row.

Il codice corrispondente in ANSI C è:

```
nome_variabile = nome_vettore[numer_intero/nome_variabile_intera];
```

esempio:

```
carattere = caratteri[0];
```

Operazioni

I seguenti blocchi si trovano tutti nella sezione “Operazioni” del menu a sinistra e offrono all'utente delle funzionalità in più (funzioni, operazioni aritmetiche).

Generazione di numeri casuali (random)



The screenshot shows a code generation interface with a red background. It contains a dropdown menu followed by '= Random (', then another dropdown menu, a comma, a third dropdown menu, and a closing parenthesis. A red 'X' icon is at the end of the row.

Il seguente blocco consente di richiamare una funzione che assegna a una variabile di tipo INT un valore pseudocasuale compreso tra due numeri. È formato da tre caselle a discesa: la prima consente di scegliere la variabile INT, la seconda di inserire o un valore in input (intero ≥ 0) o una variabile INT, la terza di inserire un valore in input (maggiore del primo inserito) o una variabile INT.

Esempio:

Si sceglie una variabile INT, e i due estremi: 10, 20 (in questo caso)

A screenshot of a software interface showing a variable assignment. A dropdown menu on the left is set to 'intero'. To its right is an equals sign followed by the text 'Random ('. Inside the parentheses, there are two more dropdown menus: the first is set to '10' and the second to '20'. The entire expression is enclosed in closing parentheses ')'. A small red 'X' icon is visible to the right of the closing parenthesis.

Il corrispondente codice in ANSI C è:

```
int main()
{
    srand(time(NULL) + getpid()); /*gestisce eventuali numeri random*/
    ...
    nome_variabile_intera=random(valore_intero/nome_variabile_intera,valore_intero/nome_variabile_intera);
}

int random (int min, int max)
{
    return rand() % (max - min + 1) + min;
}
```

esempio:

```
int main()
{
    srand(time(NULL) + getpid()); /*gestisce eventuali numeri random*/
    ...
    intero = random(10, 20);
}

int random (int min, int max)
{
    return rand() % (max - min + 1) + min;
}
```

Numeri random negativi:

Per generare numeri random anche negativi, bisogna utilizzare due variabili di tipo INT di appoggio, dato che non si possono immettere valori < 0 nell'Inputbox (solo nei blocchi di questa sezione).

Esempio completo:

A screenshot of a software interface divided into two main sections. The left section is titled 'MAIN' and contains three lines of code in a variable assignment format: 'Porta intero_ a -10', 'Porta intero_ a 10', and 'intero = Random (intero_ intero_)'. The right section is titled 'VARIABILI e VETTORI' and contains three rows, each with a 'Variabile' column (containing 'intero', 'intero_', and 'intero_') and a 'Tipo' column (all containing 'INT'). Each row has a small red 'X' icon to its right.

Radice quadrata di un numero

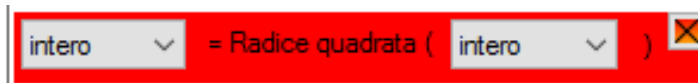
A screenshot of a software interface showing a variable assignment. A dropdown menu on the left is empty. To its right is an equals sign followed by the text 'Radice quadrata ('. Inside the parentheses, there is another empty dropdown menu. The entire expression is enclosed in closing parentheses ')'. A small red 'X' icon is visible to the right of the closing parenthesis.

Il seguente blocco consente di calcolare la radice quadrata di un numero e inserirlo all'interno di una variabile di tipo INT o DOUBLE. È composto da due caselle a discesa: la prima consente di

scegliere la variabile INT, la seconda consente di inserire un valore in input (intero ≥ 0) o una variabile intera (radicando).

Esempio:

Si sceglie una variabile INT e si assegna ad essa il suo valore sotto radice



Il corrispondente codice in ANSI C è:

```
Nome_variabile_intera/double = sqrt(valore_intero/nome_variabile_intera);
```

esempio:

```
intero = sqrt(intero);
```

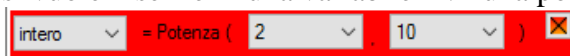
Potenza di un numero



Il seguente blocco consente di assegnare ad una variabile di tipo INT un numero creato attraverso le potenze aritmetiche. È formato da tre caselle a discesa: la prima consente di scegliere la variabile INT, la seconda consente di scegliere la base della potenza: valore numerico (intero ≥ 0) o una variabile INT, la terza consente di scegliere l'esponente della potenza: valore numerico (intero ≥ 0) o una variabile INT.

Esempio:

si vuole inserire in una variabile INT una potenza: 2^{10}



Il corrispondente codice in ANSI C è:

```
Nome_variabile_intera=pow(valore_intero/nome_variabile_intera, valore_intero/nome_variabile_intera);
```

esempio:

```
intero = pow(2,10);  
intero = pow(intero,10);
```

Utilizzo di basi e/o esponenti negativi:

Dato che non si possono inserire dei valori interi negativi (precedentemente detto) allora per avere basi e/o esponenziali con valore negativo bisogna utilizzare variabili di appoggio.

Esempio completo:

MAIN				VARIABILI e VETTORI	
Porta	base	a	10	Variable	intero
Porta	esponente	a	-2	Tipo	INT
intero	= Potenza (base	esponente	Variable	base
)			Tipo	INT
				Variable	esponente
				Tipo	INT

Controlli

I seguenti blocchi si trovano tutti nella sezione “Controlli” del menu a sinistra e consentono di effettuare controlli sulle variabili, aggiungere oggetti extra per input di dati (casella di testo, numeri, caratteri,...), gestire l’input e l’output dei programmi ANSI C.

Casella di testo

La seguente casella di testo può essere utilizzata per inserire valori di tipo STRING all’interno di altri blocchi che più avanti verranno citati. La casella accetta qualsiasi carattere della tastiera tranne i tre caratteri: “ (doppi apici), ‘ (apicetti) e ‘\$’, perché vengono utilizzati dal programma per altri fini.

Casella di carattere

La seguente casella, ha le stesse funzionalità di una casella di testo normale ma è possibile scrivere al suo interno, al massimo un solo valore. Anche quest’ultima non accetta i tre caratteri citati precedentemente.

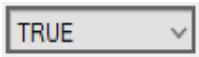
Casella dei numeri interi

La seguente casella permette di inserire valori numerici interi. Quindi i soli caratteri accettati da tastiera sono: 0,1,2,3,4,5,6,7,8,9.

Casella dei numeri reali (double)

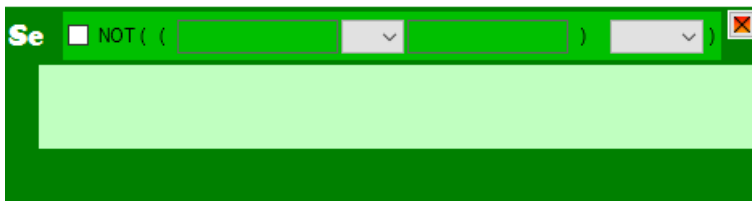
La seguente casella permette di inserire valori numerici reali. Quindi i soli caratteri accettati da tastiera sono: 0,1,2,3,4,5,6,7,8,9, e il ‘.’ (punto) che ne può essere inserito uno solo all’interno di una casella o nessuno.

Casella a discesa per i valori booleani



La seguente casella a discesa permette di scegliere tra TRUE o FALSE che descrivono i possibili valori che una variabile booleana può contenere. Viene utilizzata esclusivamente per i controlli (if, ifelse, cicli while, cicli do while).

Controllo if



Il seguente blocco consente di effettuare controlli sulle variabili semplici (non i vettori). È formato da un blocco che rappresenta la condizione (bloccoControllo) e da un pannello, dove è possibile, attraverso il Drag&Drop, inserire altri blocchi. Il codice inserito all'interno del pannello verrà eseguito solo se la condizione inserita dall'utente è verificata.

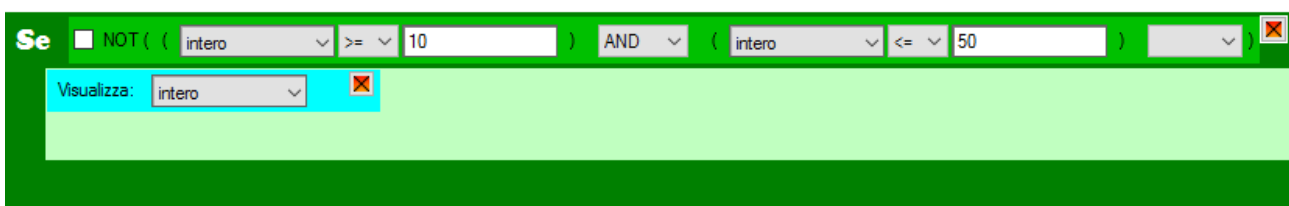
Il bloccoControllo è formato da varie parti:

1. Una checkbox, che se spuntata, rappresenta la negazione dell'intera condizione (nega tutto).
2. Due pannelli droppabili: Si possono inserire al suo interno caselle di testo, caselle di carattere, caselle numeriche intere o reali, caselle a discesa per le variabili o i valori booleani. Grazie a questi si posso fare i vari controlli. I due oggetti inseriti nei due pannelli per il confronto devono essere compatibili, cioè i loro tipi devono corrispondere. Per esempio se si inserisce una Combobox e si sceglie una variabile di tipo STRING, non si può confrontarla con una casella dei numeri interi, perché altrimenti il file ANSI C non verrà eseguito. Invece è possibile inserire una casella di testo o un'altra Combobox scegliendo un'altra variabile STRING.
3. Due caselle a discesa: la prima consente di scegliere tra i vari operatori di confronto: (==, !=, <=, >=, <, >), la seconda consente di scegliere tra AND e OR. Se si sceglie tra questi due valori, verrà inserito in coda all'oggetto bloccoControllo un altro oggetto identico a lui, ma senza la checkbox (per gestire condizioni che si devono verificare contemporaneamente).

Il bloccoControllo viene utilizzato anche nei blocchi ifelse, ciclo while e ciclo dowhile, con la stessa funzionalità.

Esempio:

Si vuole verificare che una variabile di tipo INT abbia un valore compreso tra il 10 e il 50. Se sì, allora si stampa il contenuto della variabile.



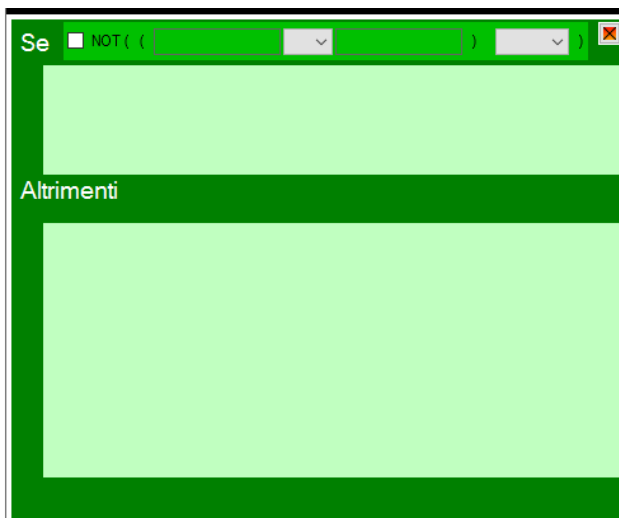
Il corrispondente codice in ANSI C è:

```
if([!](elemento condizione elemento) operatore_&&_|| (...))  
{  
    ...  
}
```

esempio:

```
if((intero >= 10) && (intero <= 50))  
{  
    ...  
}
```

Controllo if else



Il seguente blocco consente, anche lui, di effettuare controlli sulle variabili semplici (non i vettori). È formato da un blocco che rappresenta la condizione (bloccoControllo) e da due pannelli, dove è possibile, attraverso il Drag&Drop, inserire altri blocchi. Il codice inserito all'interno del primo pannello verrà eseguito solo se la condizione inserita dall'utente è verificata, altrimenti verrà eseguito quello contenuto nel secondo pannello (else).

Esempio:

Si vuole verificare che una variabile INT, con valore casuale tra 0 e 100, contenga un valore pari oppure dispari e stampare il risultato.

The image shows a visual programming interface with the following components:

- A red block: `intero` = Random (`0` , `100`)
- An orange block: `resto` = `intero` % `2`
- A green conditional block starting with "Se" (If):
 - Condition: NOT (`resto` == `0`)
 - Then branch (light green background):
 - Visualizza: `Pari\n`
 - Altrimenti branch (dark green background):
 - Visualizza: `Dispari\n`

Il corrispondente codice in ANSI C è:

```
if(!((elemento condizione elemento) operatore_&&_|| (...)))  
{  
    ...  
}  
else  
{  
    ...  
}
```

esempio:

```
if((resto == 0))  
{  
    ...  
}  
else  
{  
    ...  
}
```

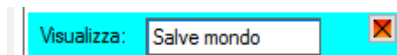
Stampa a video di un testo/variabile

The image shows a single block with the label "Visualizza:" followed by an empty text input field and a small icon with a red 'X' in a blue square.

Il seguente blocco consente di stampare sulla videata della console un testo o un valore di una variabile. È costituito da un pannello draggabile che accetta solo caselle di testo o caselle a discesa che contengono le varie variabili.

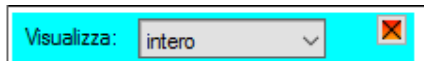
Esempio:

Si vuole visualizzare un testo: “Salve mondo”



Se si vuole andare a capo si può inserire nel testo '\n' che in ANSI C consente di andare a capo nella console.

Se si vuole stampare una variabile INT:



Il corrispondente codice in ANSI C è:

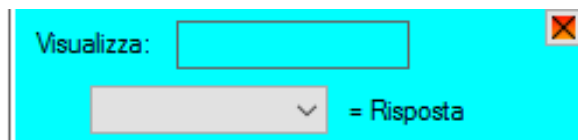
```
printf("%tipo_dato", "valore"/variabile);
```

dove il tipo di dato è: %s (STRING), %d (INT), %lf (DOUBLE), %s (BOOL, perché sono visualizzati attraverso una if), %c (CHAR)

esempio:

```
printf("%s", "Salve mondo");  
printf("%d", intero);
```

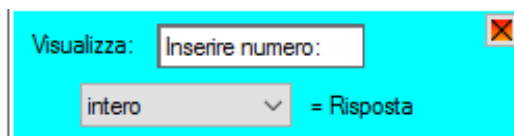
Input di una variabile



Il seguente blocco gestisce l'input delle variabili. Consente all'utente di inserire, a programma avviato, dei valori che verranno immessi all'interno di variabili. È costituito da un oggetto visualizzaTesto e da una Combobox che consente di scegliere tra le variabili INT, STRING, CHAR, DOUBLE. In base al tipo di variabile si possono inserire valori differenti: INT (solo numeri interi), DOUBLE (numeri reali con o senza punto), CHAR (un solo carattere), STRING (una stringa. Dato che, però, viene utilizzata la scanf, non si possono immettere frasi utilizzando gli spazi ma solo una parola intera). Se il valore viene immesso correttamente allora si proseguirà con il programma, altrimenti il file *.exe terminerà.

Esempio:

Si vuole chiedere in input un numero e inserirlo in una variabile INT



Il corrispondente codice in ANSI C è:

```
scanf("%tipo_dato", [&]variabile);
```

esempio:

```
scanf("%d", &intero);
```

Cicli

I seguenti blocchi si trovano tutti nella sezione “Cicli” del menu a sinistra e consentono l’esecuzione di altri blocchi in modo iterativo utilizzando le tre tipologie di cicli.

Ciclo precondizionale (while)

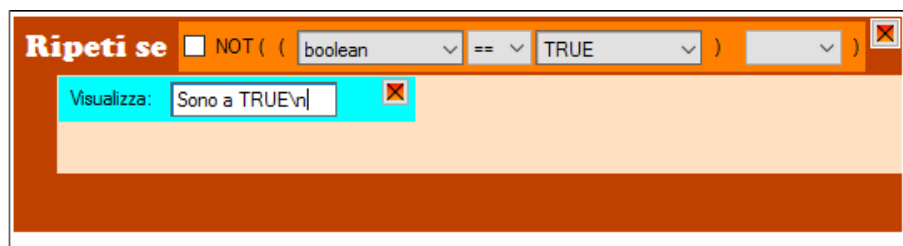


Il seguente blocco consente di eseguire delle istruzioni in modo iterativo (ripetuto) se viene verificata la condizione che viene inserita dall’utente. I blocchi possono essere inseriti attraverso il Drag&Drop all’interno del pannello.

Il precondizionale, potrebbe, anche, mai eseguire il codice inserito al suo interno, se la condizione non è verificata all’inizio della sua esecuzione.

Esempio:

Si vuole eseguire un ciclo fino a che non si ha una variabile BOOL a TRUE e stampare il testo “Sono a TRUE” e andare a capo ogni volta.



Il corrispondente codice in ANSI C è:

```
while(condizione)
{
    ...
}
```

esempio:

```
while(boolean == TRUE)
{
    ...
}
```

Ciclo postcondizionale (do while)



Il seguente blocco consente di eseguire delle istruzioni in modo iterativo se viene verificata la condizione che viene inserita dall'utente (in fondo al blocco). I blocchi possono essere inseriti attraverso il Drag&Drop all'interno del pannello.

Il postcondizionale, esegue il codice inserito al suo interno una volta prima di fare il primo controllo.

Esempio:

Si deve eseguire un ciclo che incrementi di 1 una variabile INT. Quando arriva al valore 10 allora il ciclo smette e si stampa il valore della variabile.



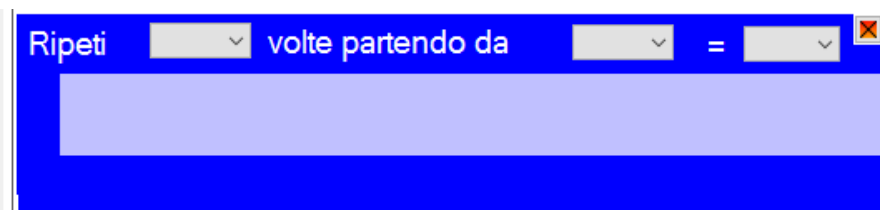
Il corrispondente codice in ANSI C è:

```
do
{
...
}while(condizione)
```

esempio:

```
do
{
...
}while(intero < 10)
```

Ciclo con il contatore (for)



Il seguente blocco consente di eseguire delle istruzioni in modo iterativo per un numero fisso di volte. I blocchi possono essere inseriti attraverso il Drag&Drop all'interno del pannello.

Il ciclo con il contatore, esegue il codice inserito per n volte, quindi viene utilizzato quando si sa già per quante volte si devono eseguire le istruzioni del ciclo. Il blocco è formato da tre caselle a discesa: la prima permette di scegliere il numero di ripetizioni (numero intero o variabile INT), la seconda permette di scegliere la variabile INT che servirà da contatore per il ciclo, la terza consente di scegliere il numero di partenza della variabile contatore (numero intero o variabile INT).

Esempio:

Si vuole caricare un vettore di 10 elementi INT, stamparli uno per uno e sommarli tra di loro e visualizzare la somma alla fine del ciclo:



Il corrispondente codice in ANSI C è:

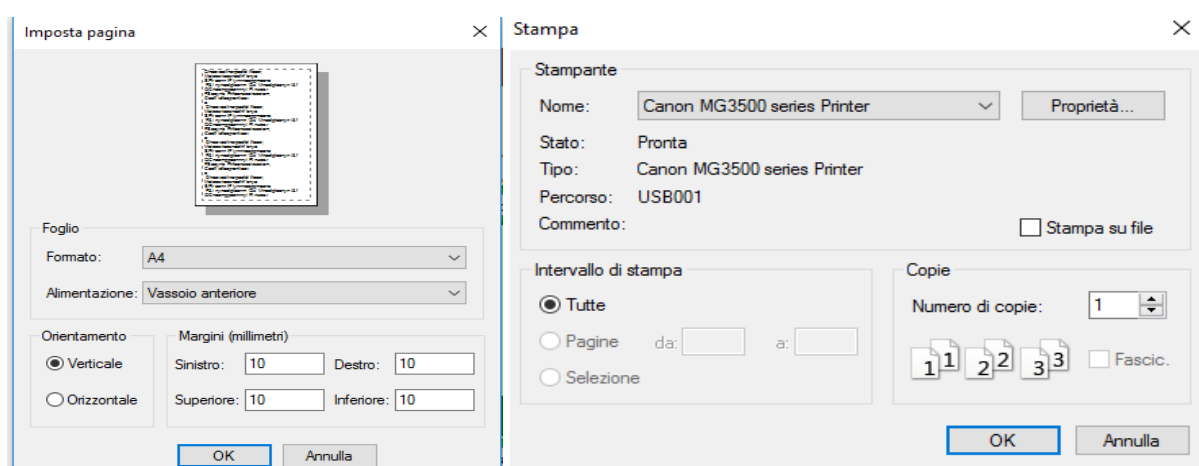
```
for(nome_variabile_intera = valore_intero/nome_variabile_intera; nome_variabile_intera <
    valore_intero/variabile_intera_condizione; nome_variabile_intera++)
{
    ...
}
```

esempio:

```
for(i = 0; i < 10; i++)
{
    ...
}
```

Stampa di un progetto (file ANSI C)

Il programma permette anche la stampa e la creazione del file PDF del file ANSI C. Per fare questo, bisogna selezionare la voce “Stampa->file C” dal sottomenù “File”. Il programma avvierà la stampa e salverà il PDF nella cartella del progetto, dopo che l’utente avrà scelto le sue preferenze dalle finestre che compariranno:



Esempi di progetti Blocco C

Gioco dell'alto basso

Il seguente progetto simula il gioco dell'alto basso: il computer genera un numero casuale tra 0 e 100, e l'utente deve cercare di indovinarlo nel minor numero di tentativi, inserendo un numero in input. Il programma verificherà se ha indovinato o se il numero è troppo basso o troppo alto e lo riferirà all'utente.

Codice in Blocco C:

MAIN	VARIABILI e VETTORI												
<pre> numero_s Random (0 100) Porta Contatore a 0 </pre>	<table border="1"> <tr> <td>Variable</td> <td>numero_seg</td> <td>Tipo</td> <td>INT</td> </tr> <tr> <td>Variable</td> <td>Tentativo</td> <td>Tipo</td> <td>INT</td> </tr> <tr> <td>Variable</td> <td>Contatore</td> <td>Tipo</td> <td>INT</td> </tr> </table>	Variable	numero_seg	Tipo	INT	Variable	Tentativo	Tipo	INT	Variable	Contatore	Tipo	INT
Variable	numero_seg	Tipo	INT										
Variable	Tentativo	Tipo	INT										
Variable	Contatore	Tipo	INT										
<p>Ripeti</p> <pre> Visualizza: Inserire il tentativo: Tentativo = Risposta Contatore + 1 Se NOT (Tentativo == numero_segreto) Visualizza: Hai indovinato!!!\n Visualizza: In Visualizza: Contatore Visualizza: tentativi. Visualizza: \n Altrimenti Se NOT (Tentativo > numero_segreto) Visualizza: ALTO\n Altrimenti Visualizza: BASSO\n </pre>													
<pre> Se NOT (Tentativo != numero_segreto) </pre> <p>FINE MAIN</p>													

Codice in ANSI C:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

int main()
{
    int numero_segreto = 0;
    int Tentativo = 0;
    int Contatore = 0;
    numero_segreto = random(0, 100);
    Contatore = 0;
    do
    {
        printf("%s", "Inserire il tentativo: ");
        scanf("%d", &Tentativo);
        getchar();
        Contatore = Contatore + 1;
        if((Tentativo == numero_segreto))
        {
            printf("%s", "Hai indovitato!!!\n");
            printf("%s", "In ");
            printf("%d", Contatore);
            printf("%s", " tentativi.");
            printf("%s", "\n");
        }
        else
        {
            if((Tentativo > numero_segreto))
            {
                printf("%s", "ALTO\n");
            }
            else
            {
                printf("%s", "BASSO\n");
            }
        }
    }while((Tentativo != numero_segreto));
    printf("Premi un pulsante per uscire...");
    getchar();
    return 0;
}

int random (int min, int max)
{
    return rand() % (max - min + 1) + min;
}
```

Esempio di esecuzione:



```
Inserire C:\Users\Fiore\Documents\PROGETTO_FINALE\ese02_altoBasso\ese02_altoBasso.exe
Inserire il tentativo: 12
BASSO
Inserire il tentativo: 45
BASSO
Inserire il tentativo: 58
ALTO
Inserire il tentativo: 50
BASSO
Inserire il tentativo: 55
BASSO
Inserire il tentativo: 56
BASSO
Inserire il tentativo: 57
Hai indovitato!!!
In 7 tentativi.
Premi un pulsante per uscire...
^
```

Somma degli elementi di un vettore

Il seguente progetto consente di sommare gli elementi di un vettore di lunghezza 10 e di stampare il suo valore sulla console. Ogni elemento è un numero casuale tra 0 e 10.

Codice in Blocco C:

The screenshot shows a block-based programming interface with two main panels: **MAIN** and **VARIABILI e VETTORI**.

MAIN Panel: Contains a sequence of blocks for program logic. It starts with a 'Porta' block setting 'somma' to 0. Then, a 'lunghezza' block calculates the length of the 'vettore' array. A 'Ripeti' loop block is configured to run 10 times, starting from 0. Inside the loop, an 'app' block generates a random number between 0 and 10. This value is then assigned to 'vettore[i]' and 'app_due'. The 'app_due' value is printed, and 'somma' is incremented by 'app_due'. After the loop, 'somma' is printed, and a final '\n' is printed.

VARIABILI e VETTORI Panel: Lists the variables and vectors used in the program. It includes:

- Variable 'somma' of type INT.
- Variable 'lunghezza' of type INT.
- Vector 'vettore' of type INT, length 10.
- Variable 'i' of type INT.
- Variable 'app' of type INT.
- Variable 'app_due' of type INT.

The interface ends with a **FINE MAIN** block.

Codice in ANSI C:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define lunghezzaVet(array) (sizeof(array)/sizeof(array[0])) //calcola la lunghezza del
vettore

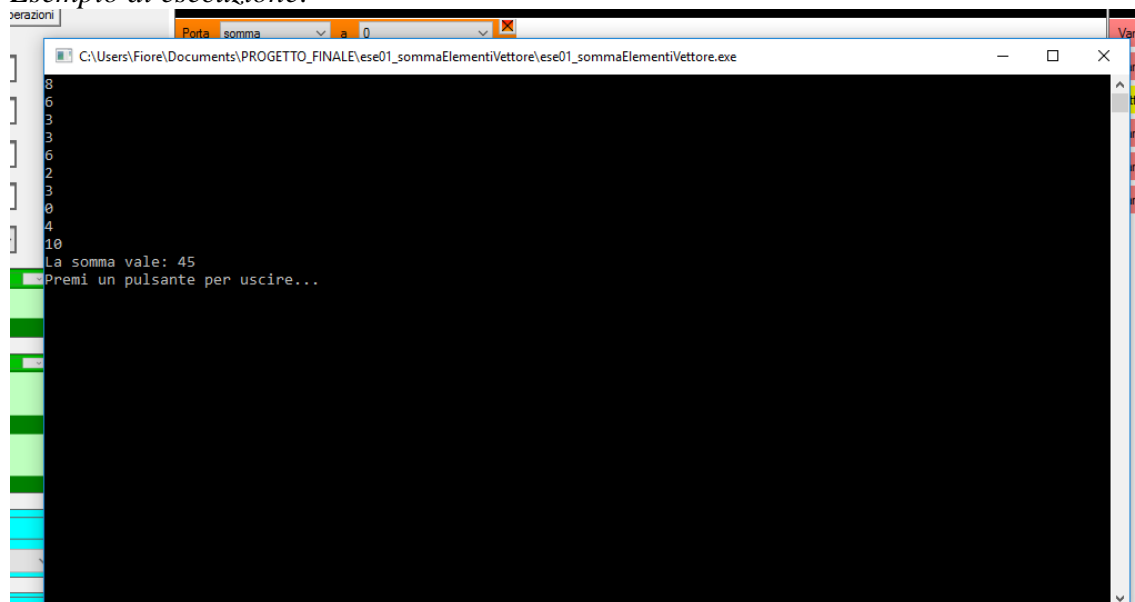
int main()
{
    srand(time(NULL) + getpid()); /*gestisce eventuali numeri random*/
    int somma = 0;
    int lunghezza = 0;
    int vettore[10];
    int i = 0;
    int app = 0;
    int app_due = 0;
    somma = 0;
    lunghezza = lunghezzaVet(vettore);

    for(i = 0; i < lunghezza; i++ )
    {
        app = random(0, 10);
        vettore[i] = app;
        app_due = vettore[i];
        printf("%d", app_due);
        somma = somma + app_due;
        printf("%s", "\n");
    }
    printf("%s", "La somma vale: ");
    printf("%d", somma);
    printf("%s", "\n");
}
```

```
printf("Premi un pulsante per uscire...");
getchar();
return 0;
}

int random (int min, int max)
{
    return rand() % (max - min + 1) + min;
}
```

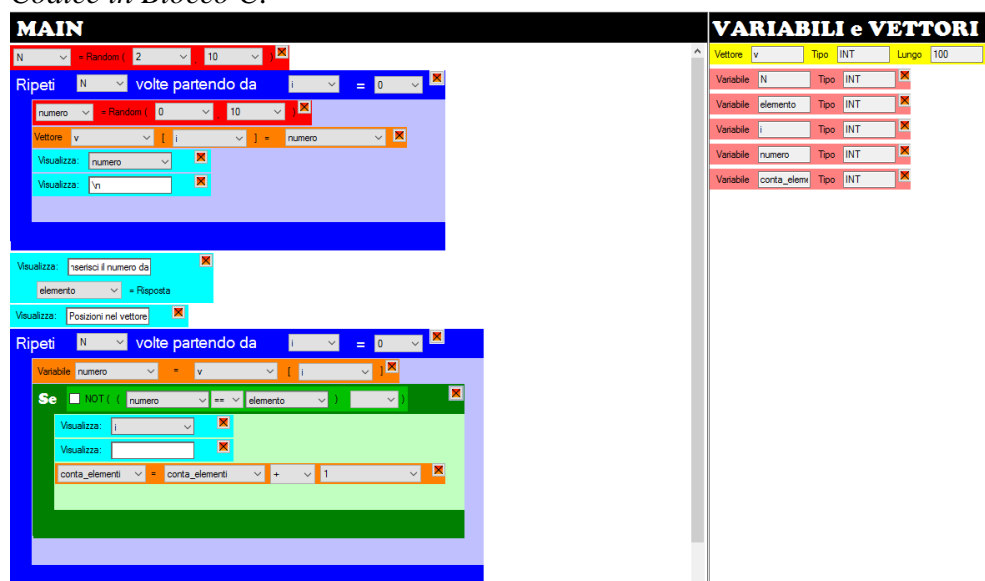
Esempio di esecuzione:



Ricerca di elemento all'interno di un vettore con elementi disordinati e ripetuti

Il seguente progetto consente di ricercare un elemento, dopo che l'utente l'ha inserito in input, all'interno di un vettore di N numeri interi. Se il valore è contenuto all'interno del vettore, il programma stampa tutte le posizioni dell'elemento, altrimenti stampa un messaggio negativo: *"Elemento non contenuto nel vettore!"*.

Codice in Blocco C:





Codice in ANSI C:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

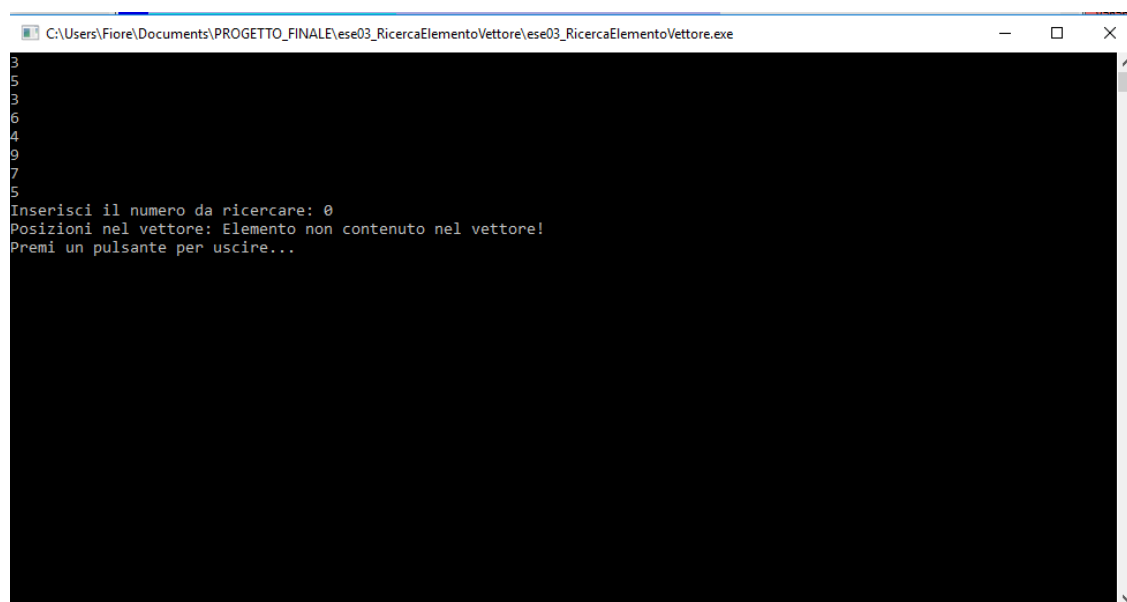
int main()
{
    int v[100];
    int N = 0;
    int elemento = 0;
    int i = 0;
    int numero = 0;
    int conta_elementi = 0;
    N = random(2, 10);
    for(i = 0; i < N; i++ )
    {
        numero = random(0, 10);
        v[i] = numero;
        printf("%d", numero);
        printf("%s", "\n");
    }
    printf("%s", "Inserisci il numero da ricercare: ");
    scanf("%d", &elemento);
    getchar();
    printf("%s", "Posizioni nel vettore: ");
    for(i = 0; i < N; i++ )
    {
        numero = v[i];
        if((numero == elemento))
        {
            printf("%d", i);
            printf("%s", " ");
            conta_elementi = conta_elementi + 1;
        }
    }
    if((conta_elementi == 0))
    {
        printf("%s", "Elemento non contenuto nel vettore!");
    }
    printf("%s", "\n");
    printf("Premi un pulsante per uscire...");
    getchar();
    return 0;
}

int random (int min, int max)
{
    return rand() % (max - min + 1) + min;
}
```

Esempio di esecuzione:



```
C:\Users\Fiore\Documents\PROGETTO_FINALE\ese03_RicercaElementoVettore\ese03_RicercaElementoVettore.exe
4
5
10
3
5
5
9
Inserisci il numero da ricercare: 5
Posizioni nel vettore: 1 4 5
Premi un pulsante per uscire...
```



```
C:\Users\Fiore\Documents\PROGETTO_FINALE\ese03_RicercaElementoVettore\ese03_RicercaElementoVettore.exe
3
5
3
6
4
9
7
5
Inserisci il numero da ricercare: 0
Posizioni nel vettore: Elemento non contenuto nel vettore!
Premi un pulsante per uscire...
```

Ordinamento degli elementi di un vettore di caratteri (metodo per selezione)

Il seguente progetto permette di caricare un vettore di 10 caratteri, facendoli immettere dall'utente, e poi di stamparlo in modo crescente (dal più piccolo al più grande), utilizzando l'algoritmo fondamentale di ordinamento per selezione.

Codice in Blocco C:

MAIN

lunghezz = Lunghezza (v)

Ripeti lunghez volte partendo da i = 0

Visualizza: Inserire un carattere

c = Risposta

Vettore v [i] = c

N = lunghezza - 2

Ripeti N volte partendo da i = 0

Porta min a i

Ripeti lunghez volte partendo da i = 1

Variable c = v [i]

Variable v_min = v [min]

Se NOT (v_min > c)

Porta min a i

Se NOT (min == i)

Variable aus = v [i]

Variable v_min = v [min]

Vettore v [i] = v_min

Vettore v [min] = aus

Ripeti lunghez volte partendo da i = 0

Variable c = v [i]

Visualizza: c

Visualizza: \n

FINE MAIN

VARIABILI e VETTORI

Variable	Tipo	Lungo
v	CHAR	10
i	INT	
lunghezza	INT	
N	INT	
min	INT	
aus	CHAR	
j	INT	
v_min	CHAR	

Codice in ANSI C:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define lunghezzaVet(array) (sizeof(array)/sizeof(array[0])) //calcola la lunghezza del vettore

int main()
{
    char c = ' ';
    char v[10];
    int i = 0;
    int lunghezza = 0;
    int N = 0;
    int min = 0;
```

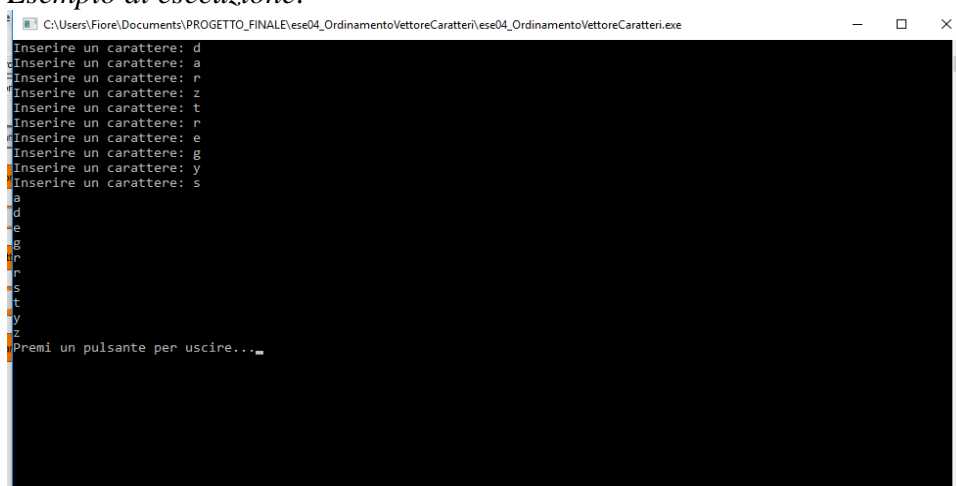


```
char aus = ' ';
int j = 0;
char v_min = ' ';
lunghezza = lunghezzaVet(v);

for(i = 0; i < lunghezza; i++ )
{
    printf("%s", "Inserire un carattere: ");
    scanf("%c", &c);
    getchar();
    v[i] = c;
}
N = lunghezza - 2;
for(i = 0; i < N; i++ )
{
    min = i;
    for(j = i; j < lunghezza; j++ )
    {
        c = v[j];
        v_min = v[min];
        if((v_min > c))
        {
            min = j;
        }
    }
    if(!(min == i))
    {
        aus = v[i];
        v_min = v[min];
        v[i] = v_min;
        v[min] = aus;
    }
}
for(i = 0; i < lunghezza; i++ )
{
    c = v[i];
    printf("%c", c);
    printf("%s", "\n");
}
printf("Premi un pulsante per uscire...");
getchar();
return 0;
}

int random (int min, int max)
{
    return rand() % (max - min + 1) + min;
}
```

Esempio di esecuzione:

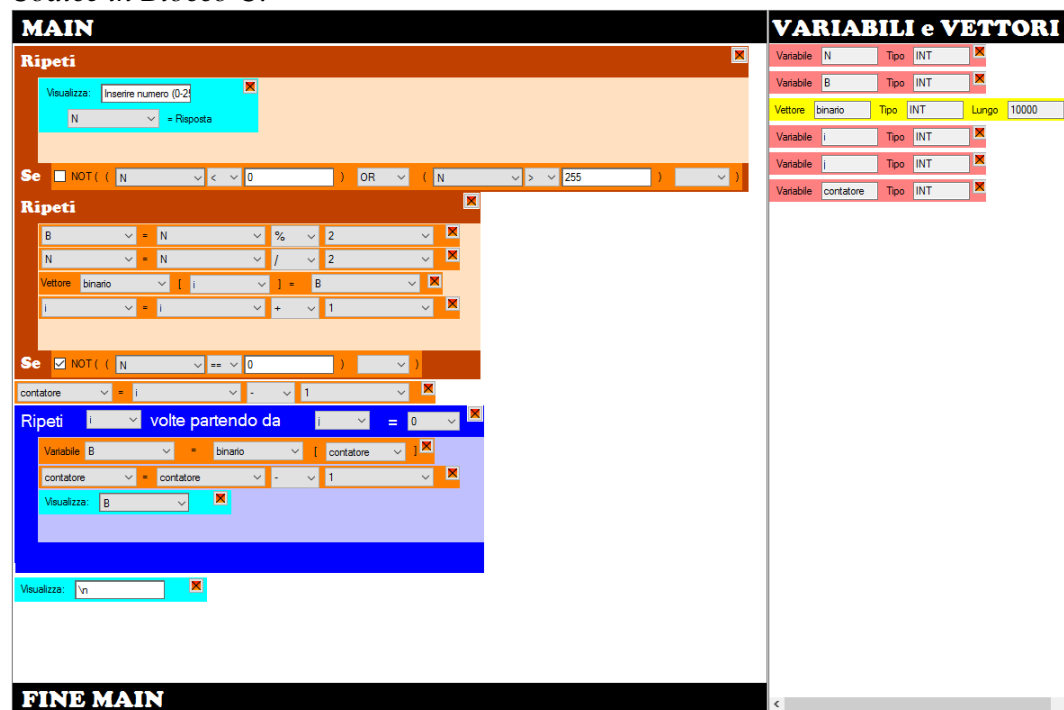


```
C:\Users\Fiore\Documents\PROGETTO_FINALE\ese04_OrdinamentoVettoreCaratteri\ese04_OrdinamentoVettoreCaratteri.exe
Inserire un carattere: d
Inserire un carattere: a
Inserire un carattere: r
Inserire un carattere: z
Inserire un carattere: t
Inserire un carattere: r
Inserire un carattere: e
Inserire un carattere: g
Inserire un carattere: y
Inserire un carattere: s
Premi un pulsante per uscire...
```

Convertitore di numeri da decimale a binario

Il seguente progetto permette, dopo che l'utente ha inserito un numero intero positivo compreso tra 0 e 255 in input, di convertirlo nel corrispondente valore in binario e stamparlo a video.

Codice in Blocco C:



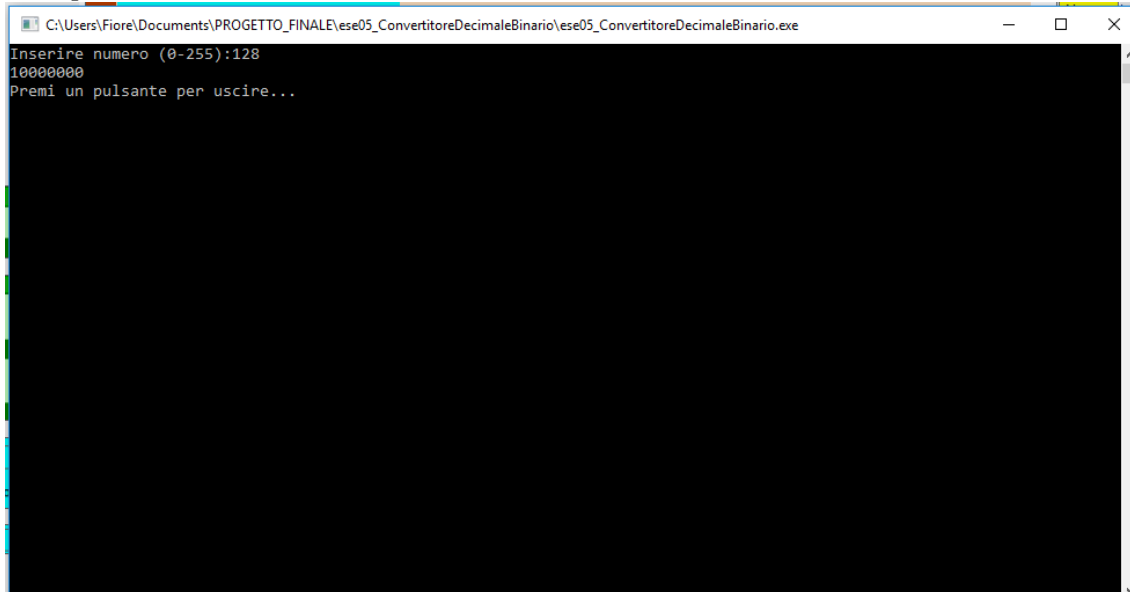
Codice in ANSI C:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

int main()
{
    srand(time(NULL) + getpid()); /*gestisce eventuali numeri random*/
    int N = 0;
    int B = 0;
    int binario[10000];
    int i = 0;
    int j = 0;
    int contatore = 0;
    do
    {
        printf("%s", "Inserire numero (0-255):");
        scanf("%d", &N);
        getchar();
    }while((N < 0) || (N > 255));
    do
    {
        B = N % 2;
        N = N / 2;
        binario[i] = B;
        i = i + 1;
    }while(!(N == 0));
    contatore = i - 1;
    for(j = 0; j < i; j++ )
    {
        B = binario[contatore];
        contatore = contatore - 1;
    }
}
```

```
        printf("%d", B);  
    }  
    printf("%s", "\n");  
    printf("Premi un pulsante per uscire...");  
    getchar();  
    return 0;  
}  
  
int random (int min, int max)  
{  
    return rand() % (max - min + 1) + min;  
}
```

Esempio di esecuzione:



```
C:\Users\Fiore\Documents\PROGETTO_FINALE\ese05_ConvertitoreDecimaleBinario\ese05_ConvertitoreDecimaleBinario.exe  
Inserire numero (0-255):128  
10000000  
Premi un pulsante per uscire...
```