1 OPIS PROGRAMU

Program „Minimisation 2.1” służy do minimalizacji funkcji logicznych metodą ekspansji w sposób zachłanny. To znaczy, że ten program nie gwarantuje rozwiązania najlepszego.

Jako dane wejściowe program przyjmuje dwa wektory liczb w systemie dziesiętnym: zbiór prawdy i zbiór nieprawdy. Program również daje możliwość wprowadzić jako jeden z wektorów zbiór wartości nieokreślonych, wtedy pozostałe wartości zostaną wygenerowane automatycznie, lub od razu automatycznie wygenerować wektor na podstawie drugiego.

Na wyjściu program wypisuje funkcji zminimalizowane i proces minimalizacji. Istnieje możliwość zapisywania wyniku do pliku.

2 OPIS DZIAŁANIA PROGRAMU

2.1 Opis danych

Nazwa typu danych	Pola	Opis
enum matrix_type {F, R, Z, D}	Typ macierzy: F – macierz jedynek, R – macierz zer, Z – wypełnić na podstawie drugiej macierzy, D – macierz wartości nieokreślonych.	
vector_with_indikator	vector<int> vec	Wektor wejściowych wartości.
	matrix_type indikator	Typ zbioru wejściowego.
matrix_b	matrix matr	Kostka.
	vector<vector<int> > implicants	Implikanty z tej kostki.
vec_of_impl	int num_of_line_from_F	Numer kolumny zbioru implikant w macierzy jedynkowej.
	vector<int> vec	Zbiór implikant z kostki.
matrix	vector<QBitArray> array	Tablica dwuwymiarowa, składająca się z wektora tablic bitowch. Wykorzystanie tablic bitowych oszczędza pamięć, używaną przez macierze do 64 razy odnośnie tablic integerów.

	int numOfRows	Liczba wierszy macierzy.
	int numOfColumns	Liczba kolumn macierzy.

2.2 Opis funkcji

Prototyp funkcji	Opis
bool **cmp_for_same_vec_of_impl** (vec_of_impl *A*, vec_of_impl *B*)	Zwraca true, jeżeli wektor *A* jest równy wektorowi *B*.
bool **cmp_vec_of_impl** (vec_of_impl *A*, vec_of_impl *B*)	Zwraca true, jeżeli wektor *A* jest mniejszy od wektora *B*.
bool **current_b_has_this_implicant** (const vector<vec_of_impl> *&vector_of_implicants*, const vector<matrix_b> *&vector_of_b*, int *row*, int *column*)	Zwraca true, jeżeli kostka o indeksie *row* z *vector_of_implicants* zawiera implikantę, która znajduje się w *vector_of_b* pod indeksem *column*.
bool **vectors_have_collision** (vector_with_indikator *&vec1*, vector_with_indikator *&vec2*)	Zwraca true, jeżeli dwa podanych wektora mają wspólne wartości.
int **invert** (const int *val*)	Zwraca 1, jeżeli podana wartość jest równa 0 i odwrotnie.
matrix **createMatrix** (int *numOfLines*, int *numOfColumns*)	Tworzy macierz o *numOfLines* wierszach i *numOfColumns* kolumnach.
matrix **get_result_matrix** (const vector<vec_of_impl> *&vector_of_implicants*, const vector<matrix> *&fr*, const vector<matrix_b> *&vector_of_b*)	Zwraca macierz końcową implikant, zbudowaną na postawie wejściowych parametrów(algorytm 'get_result_matrix').
matrix **make_bin_matrix_by_dec_vec** (const vector<int> *&vec*, int *max*)	Zwraca macierz, wypełnioną wartościami z wektora *vec* w postaci dwójkowej. *max* To maksymalna wartość z dwóch wektorów wejściowych.
string **print_implicant_as_string** (vector<matrix_b> *mat_b*, int *n_line*, int *j*, vector<matrix> *fr*)	Zwraca napis z implkantą *j* z linii *n_line* kostki *mat_b*.
string **show_matrix_as_string** (const matrix *&array*)	Zwraca wypisaną macierz *array* w postaci napisu.

vector<int> **make_by_other** (const vector<int> *&vec1*, const vector<int> *&vec2*)	Tworzy wektor wartości od 0 do maksymalnej wartości z *vec1* i *vec2*, nie włączając do niego wartości, zawartych w tych wektorach na podstawie algorytmu 'make_by_other'.
vector<int> **make_by_other** (const vector<int> *&vec1*)	Tworzy wektor wartości od 0 do maksymalnej wartości z *vec1*, nie włączając do niego wartości, zawartych w tym wektorze na podstawie algorytmu 'make_by_other'.
vector<int> **toBinary** (int *deci*)	Zwraca liczbę *deci* w postaci dwójkowej jako wektor zero-jedynkowy.
vector<matrix> **get_f_and_r** (vector_with_indikator *&vec1*, vector_with_indikator *&vec2*)	Zwraca wektor, zawierający macierzy jedynek i zer, pobudowane na podstawie wektorów wejściowych (algorytm 'get_f_and_r').
vector<matrix> **get_matrix_b** (const vector<matrix> *&fr*)	Zwraca wektor kostek dla tablicy prawdy, podaną jako parametr wejściowy (algorytm 'get_matrix_b').
vector<string> **get_finally_functions** (const vector<matrix> *&fr*, const matrix *&result_matr*, vector<vec_of_impl> *&vector_of_implicants*)	Zwraca wektor napisów, każdy z których jest jedną z funkcji wynikowych, pobudowane na podstawie wejściowych parametrów(algorytm 'get_finally_functions').
vector<vec_of_impl> **get_vector_of_implicants** (IN const vector<matrix> *&fr*, OUT vector<matrix_b> *&mat_b*)	Zwraca wektor numerów minimalnych implikant z macierzy funkcji o tablicę prawdy *fr* i zapisuje do *mat_b* kostki.
vector<vector<int> > **cross_out_implicants** (vector<vector<int> > *in_ones*)	Zwraca wektor implikant *in_ones* z wykreślonymi implikantami, niepotrzebnymi do wyznaczenia minimalnego pokrycia kolumnowego(algorytm 'cross_out_implicants').
vector<vector<int> > **get_ones_positions_from_matrix** (const matrix *&B*)	Zwraca wektor, zawierający numery kolumn, które mają wartość równą 1 dla każdego wierszu macierzy *B* ('algorytm get_ones_positions_from_matrix').

vector<vector<int> > **only_minimal_implicants** (vector<vector<int> > *in_im*)	Zwraca wektor implikant minimalnych z wektora wejściowego.
vector<vector<int> > **open_brackets** (vector<vector<int> > *in_ones*)	Zwraca wektor implikant in_ones po otwarciu nawiasów, to znaczy przekształca funkcji z wektora do postaci dysjunkcji (algorytm 'open_brackets').
void **print_implicant** (vector<matrix_b> *mat_b*, int *n_line*, int *j*, vector<matrix> *fr*)	Wypisuje implkantę *j* z linii *n_line* kostki *mat_b*.
void **show_finally_functions** (const vector<int> *&vec1*, const vector<int> *&vec2*, char **chose*)	Wypisuje wynik działania programu dla wejściowych danych do standardowego potoku wyjścia.
void **showMatrix** (const matrix *&array*)	Wypisuje macierz *array* do standardowego potoku wyjścia.
void **write_to_file** (const string *&in_string*, const char **filename*)	Zapisuje *in_string* do pliku o nazwie *filename.*
void **write_to_file** (const vector<int> *&vec1*, const vector<int> *&vec2*, char **chose*, const char **filename*)	Zapisuje wynik działania programu dla wejściowych danych do pliku o nazwie *filename.*