**Functional Overview for Parsing**

|  |  |
| --- | --- |
| Function\_Name | Function\_Working |
| GetOneStructure (runichi2.c--L126)  ReadTheStructure (runichi2.c--L388)  CreateOrigInpDataFromMolfile (mol2atom.c--L101)  InchiToOrigAtom (runichi2.c--L1058)  ReadMolfileToInpAtoms (mol2atom.c--L307) | Gets Input of any kind like Molfile, SDF or InChI String. Uses the method **ReadTheStructure()** for reading from the input. Also, returns the method **TreatErrorsInReadTheStructure()**. In this method, length of a specific array is also adjusted accordingly to accommodate the next element.  Read’s the Input data of any kind (Molfile, SDF or InChI String). Read’s the original input structure from Molfile/SDFile. Uses the method **CreateOrigInpDataFromMolFile()** for creating the input structure. Chiral flag mode added in this method. If user uses Chiral mode or Stereo and Chiral mode simultaneously then specific conditions will run. Also, uses the method **InchiToOrigAtom()** which makes Internal molecular data from InChI string after reading the original input data as text.  Uses the method **ReadMolFileToInpAtoms()** for reading from the input file then with the help of “memcpy” copies the newly read structure to another location in memory for further processing. Also handle’s error like “Empty Structure” & “Too many atoms”.  Specifically for reading from InChI string. Uses the method **InchiToInpAtom()** for reading the input string and then populating the internal data structure with appropriate values. Also handle’s error like “Empty Structure” & “Too many atoms”. Copies the newly read information to another memory location.  Uses the method **ReadMolfile()** for reading from a MOL or SD file. Then, uses the method **MakeInpAtomsFromMolfileData()** which creates Input atoms. Deals with errors such as “Too many hydrogens at heavy atom” and “Too many atoms”. |

**Functional Overview for Conversion**

|  |  |
| --- | --- |
| Function\_Name | Function\_Working |
| ReadMolfile (mol\_fmt1.c--L97)  MolfileReadDataLines (mol\_fmt1.c--L171)  MakeInpAtomsFromMolfileData (mol2atom.c—L511) | Uses the method **MolfileReadDataLines()** for reading the data blocks in MOL or SD file. Uses the method **SDFileSkipExtraData()** for ignoring extra data present in the SD file for time being. Also uses the method **MolfileTreatPseudoElementAtoms()** (Treat star/Zz atoms).  Read’s from multiple method’s involving reading header lines, count lines (also checking to deal with V3000 Molfile), reading from atom block, bonds block, prop block. Also checking for in-range valences. The functions involved are as following: **MolfileReadHeaderLines()**, **MolfileReadCountsLine()**, **MolfileV3000ReadCTABBeginAndCountsLine()**, **MolfileReadAtomsBlock()**, **MolfileV3000ReadAtomsBlock()**, **MolfileReadBondsBlock()**, **MolfileV3000ReadBondsBlock()**, **MolfileV3000ReadTailOfCTAB()**, **MolfileReadSTextBlock()**, **MolfileReadPropBlock()**.  First of all checks if Molfile has no chemical structure then program exits otherwise input structure is created with the help of method **CreateInpAtom().** Copy atom info and bond info separately to another memory location and then fill the data structures (arrays) with the copied information. At the end valences are calculated. |

**Functional Overview for Normalization**

|  |  |
| --- | --- |
| Function\_Name | Function\_Working |
| RunBalancedNetworkSearch (ichi\_bns.c--L676)  BalancedNetworkSearch (ichi\_bns.c—L10803) | Runs the vital method **BalancedNetworkSearch()** that deals with the major Normalization procedures.  First of all, creating a base for a blossom structure with the help of method **MakeBlossom()**. Blossom structures are used as a connected component of a larger graph. A blossom structure normally handles odd structures of a graph that needs to be handled differently. Method **FindPathToVertex\_s()** tries to construct path from blossom base from a specified vertex to a base vertex. This process is done iteratively until base vertex is reached. Method **PullFlow()** is augmenting the flow along a given path from one vertex to another. Other methods involved are **AddRemoveProtonsRestr(), AddRemoveIsoProtonsRestr()**. |

**Functional Overview for Canonicalization**

|  |  |
| --- | --- |
| Function\_Name | Function\_Working |
| Canon\_INChI (ichicano.c--L2714)  Canon\_INChI3 (ichicano.c--L1356)  GetBaseCanonRanking (ichican2.c--L5059) | Runs the vital method **Canon\_INChI3()** that deals with the major Canonicalization procedures.  Basic task is to initialize the input data which is saved in data structures with the help of memory allocation for non-isotopic stereo descriptors, non-isotopic non-inverted stereo descriptors. Method **map\_stereo\_bonds4()** is responsible for mapping and optimization process and saving the resulting descriptors and numbering. Method **InvertStereo()** is used to create an initial approximation and calculate the inverted stereo descriptors for further processing. The method **FillOutStereoParities()** is responsible for filling stereo parities and then creating linear CT stereo descriptors for isotopic stereo atoms. Canon\_INChI3 also prepare’s initial ranks for the method **GetCanonRanking()**.  Starts with tautomeric or non-tautomeric processing. If tautomeric then select as a Base structure. It calls the **FillOutAtomInvariant2()** function to fill out the atom invariant for a hydrogenless skeleton. The method **SetInitialRanks2()** is used to create an equitable partition of the hydrogenless skeleton and assign initial ranks to the atoms. The method **DifferentiateRanks2()** is used to differentiate the ranks of atoms based on their connectivity and neighborhood information. The method **CanonGraph01()** determines the canonical numbering, connection table, and equivalence partition for the skeleton. The method **CanonGraph06()** performs the canonicalization process after calculating auxiliary rankings based on non-tautomeric H isotopic distribution. The method **CanonGraph07()** is used to obtain the fixed H canonical numbering, connection tables, and equivalence partition. In the end, cleaning up allocated memory and performing additional operations based on the results. |

**Functional Overview for InChI Creation**

|  |  |
| --- | --- |
| Function\_Name | Function\_Working |
| Create\_INChI (ichimake.c--L3700) | First of all, it initializes various variables and performs memory allocation for certain data structures. Then, various operations related to normalization of tautomeric and non-tautomeric structures is done. The method **mark\_alt\_bonds\_and\_taut\_groups()** is used for the normalization of possibly tautomeric structure. The method then modifies the structure and sets various properties related to tautomeric groups, alt bonds, and isotopes which are then stored in an array. Method **set\_atom\_iso\_sort\_keys()** is used to mark isotopic atoms and atoms with non-tautomeric isotopic terminal hydrogen atoms with assigning sort keys to these atoms. Method **CountTautomerGroups()** checks the atoms and bonds to identify groups that show tautomeric behavior and returns the count. Method **GetCanonLengths()** determines the canonical lengths of the tautomeric or non-tautomeric structure. Method **set\_stereo\_parity()** sets the stereo parities for the atoms in the structure. In the end, method **GetBaseCanonRanking()** is used to obtain a base canonical ranking for the structure which if successful is used for stereo canonicalization. Method **Canon\_INChI()** is used to perform actual canonicalization process which if successful is then used to fill the data structures of InChI and InChI Aux arrays. |