

☐ Description : paged to paged
☒ Claim(s) : paged to paged 5
☒ Drawing(s) : paged to paged 62
☒ Abstract : paged to paged

RTMP copy

- 1 -

Automatic program generating apparatus

The present invention relates to an automatic program generating apparatus for automatically generating a source code program without mastering a program language in program generation.

5 Conventionally, when a program is ordered to a software vendor, a customer explains the job contents to a sales engineer and requests necessary processing contents. The sales engineer explains the processing contents to a system engineer. The system engineer
10 defines program conditions for the processing contents and performs basic and detailed designs. A programmer performs coding based on the detailed design.

As described above, the process required until a source code program is generated is classified into (1)
15 a step of transmitting the processing contents from the customer to the sales engineer, (2) a step of transmitting the processing contents from the sales engineer to the system engineer, performing the basic design, and then designing the detailed portions of the
20 program based on the basic design, and (3) a step of causing the program to perform final coding. If an error (e.g., an item omission) in transmission contents is present in steps (2) and (3), although no error is present in step (1), it is difficult to find out such
25 an error due to the following reason. Since transmission of the detailed contents from the system

engineer to the program becomes time-consuming and cumbersome, the system engineer transmits only the desired conditions to the programmer. Upon reception of the desired conditions, the programmer performs coding in accordance with his/her own decision. In this case, the resultant program itself may provide a layout, prints, image display contents, all of which are different from the expected ones.

As described above, a large number of steps are required in a conventional program generating process, errors tend to occur, and time-consuming operations are required.

It is an object of the present invention to provide an automatic program generating apparatus capable of forming a program generating process as a plant-like process to uniformly generate an accurate, error-free program in accordance with an industrial technique, and allowing simple operations in the design process to shorten the program generating time.

In order to achieve the above object of the present invention, there is provided an automatic program generating apparatus comprising storage means for storing a plurality of skeletons each having a format constituted by instruction data of fixed process items for regular job contents, means for reading out a skeleton of a designated format, means for requesting to input predetermined instruction words in undefined instruction areas of the readout skeleton in a predetermined order, and program coding means for determining that necessary instruction data are input to all the undefined instruction areas of the readout skeleton and coding a resultant structure to generate a program.

According to the above means, when variable instruction words serving as undefined instruction portions are input to the skeleton (permanent instruction group) having a format satisfying the needs of the customer, a detailed program with accurate coding can be

automatically generated.

This invention can be more fully understood from the following detailed description when taken in conjunction with the accompanying drawings, in which:

5 FIG. 1 is a block diagram showing the basic arrangement of an embodiment according to the present invention;

FIG. 2 is a view for explaining a system function in FIG. 1;

10 FIGS. 3A to 3C are views for explaining an application of an apparatus according to the present invention;

FIG. 4 is a view for explaining a skeleton applied to the apparatus according to the present invention;

15 FIG. 5 is a flow chart for explaining an operation of the apparatus according to the present invention;

FIG. 6 is a view showing part of another skeleton used in the apparatus according to the present invention;

20 FIG. 7 is a view showing the remaining part of the skeleton shown in FIG. 6;

FIG. 8 is a view showing part of still another skeleton used in the apparatus according to the present invention;

25 FIG. 9 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 10 is a view showing the remaining part of the skeleton shown in FIG. 9;

30 FIG. 11 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 12 is a view showing the remaining part of the skeleton shown in FIG. 11;

35 FIG. 13 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 14 is a view showing the remaining part of the

skeleton shown in FIG. 13;

FIG. 15 is a view showing part of still another skeleton used in the apparatus according to the present invention;

5 FIG. 16 is a view showing the remaining part of the skeleton shown in FIG. 15;

FIG. 17 is a view showing part of still another skeleton used in the apparatus according to the present invention;

10 FIG. 18 is a view showing the remaining part of the skeleton shown in FIG. 17;

FIG. 19 is a view showing part of still another skeleton used in the apparatus according to the present invention;

15 FIG. 20 is a view showing the remaining part of the skeleton shown in FIG. 19;

FIG. 21 is a view showing part of still another skeleton used in the apparatus according to the present invention;

20 FIG. 22 is a view showing the remaining part of the skeleton shown in FIG. 21;

FIG. 23 is a view showing part of still another skeleton used in the apparatus according to the present invention;

25 FIG. 24 is a view showing the remaining part of the skeleton shown in FIG. 23;

FIG. 25 is a view showing part of still another skeleton used in the apparatus according to the present invention;

30 FIG. 26 is a view showing the remaining part of the skeleton shown in FIG. 25;

FIG. 27 is a view showing part of still another skeleton used in the apparatus according to the present invention;

35 FIG. 28 is a view showing the remaining part of the skeleton shown in FIG. 27;

FIG. 29 is a view showing part of still another

skeleton used in the apparatus according to the present invention;

FIG. 30 is a view showing the remaining part of the skeleton shown in FIG. 29;

5 FIG. 31 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 32 is a view showing the remaining part of the skeleton shown in FIG. 31;

10 FIG. 33 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 34 is a view showing the remaining part of the skeleton shown in FIG. 33;

15 FIG. 35 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 36 is a view showing the remaining part of the skeleton shown in FIG. 35;

20 FIG. 37 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 38 is a view showing the remaining part of the skeleton shown in FIG. 37;

25 FIG. 39 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 40 is a view showing the remaining part of the skeleton shown in FIG. 39;

30 FIG. 41 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 42 is a view showing the remaining part of the skeleton shown in FIG. 41;

35 FIG. 43 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 44 is a view showing the remaining part of the skeleton shown in FIG. 43;

5 FIG. 45 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 46 is a view showing the remaining part of the skeleton shown in FIG. 45;

10 FIG. 47 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 48 is a view showing the remaining part of the skeleton shown in FIG. 47;

15 FIG. 49 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 50 is a view showing the remaining part of the skeleton shown in FIG. 49;

20 FIG. 51 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 52 is a view showing the remaining part of the skeleton shown in FIG. 51;

25 FIG. 53 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 54 is a view showing the remaining part of the skeleton shown in FIG. 53;

30 FIG. 55 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 56 is a view showing the remaining part of the skeleton shown in FIG. 55;

35 FIG. 57 is a view showing part of still another skeleton used in the apparatus according to the present invention;

FIG. 58 is a view showing the remaining part of the skeleton shown in FIG. 57;

FIG. 59 is a view showing part of still another skeleton used in the apparatus according to the present invention;

5 FIG. 60 is a view showing the remaining part of the skeleton shown in FIG. 59;

FIG. 61 is a view showing part of still another skeleton used in the apparatus according to the present invention; and

10 FIG. 62 is a view showing the remaining part of the skeleton shown in FIG. 61.

A preferred embodiment of the present invention will be described below with reference to the accompanying drawings.

15 FIG. 1 shows the basic arrangement of the overall apparatus according to an embodiment of the present invention. A workstation 200 serving as an operation unit, a skeleton file 300, and a selection file 400 are arranged centered on a program generating section 100. A program generated using a program language in the
20 program generating section 100 is coded using an automatic coding section 500, thereby obtaining a machine language program using source codes.

The respective sections will be described below.

25 Different types of skeletons having predetermined formats and consisting of permanent instruction data corresponding to job contents are stored in correspondence with the job contents in the skeleton file 300. That is, each skeleton is a framework for generating one job program. Each skeleton consists of a step area for
30 instructions whose data processing steps are predetermined and a step area for undefined instructions. The skeleton language is, e.g., the RPG language. When a command for reading out a skeleton manu is input at the workstation 200, a display 201 in the workstation 200
35 displays a skeleton group menu. When a desired skeleton ID is designated, the program generating section 100 reads out the designated skeleton from the skeleton file

300, and the readout skeleton is stored in a work memory.

FIG. 2 shows all steps 11 of a given program X. Steps 12 surrounded by a chain line in FIG. 2 constitute a portion as a skeleton for performing a predetermined process.

A program using this skeleton 12 is unfinished, so that instruction data are filled in the undefined portions. For this purpose, the program generating section 100 requests an operator to input instruction data to the undefined portions in accordance with a predetermined order through the workstation 200. The operator 200 does not input data using a program language, but inputs data as words classified and arranged in this system using key operations. The program generating section 100 determines the input words, reads out the corresponding instruction program words (RPG) from the selection file 400, and stores the readout words in the undefined areas of the skeleton.

The selection file 400 has an internal component file 401 and a variable item file 402 because a predetermined routine may be used even in an undefined portion. The internal component file 401 stores, for example, a process for limiting the number of display characters on a screen in character display, a process for converting date data into data of the dominical year or the name of era, and the like. The variable item file 402 stores designation data for a screen layout, a slip layout, input check specifications, file update specifications, and the like.

When a variable item program is filled in one undefined area, the program generating section 100 requests the operator to fill a variable item program in the next undefined area in accordance with a predetermined priority order. In this case, the program generating section 100 sends a request upon displaying a worksheet screen having empty areas to be designated.

FIGS. 3A to 3C show a program for filing data simply input from a keyboard in a predetermined format. As shown in FIG. 3A, in a basic design, an input frame is displayed on the screen to facilitate a data input operation (step A1). A key input process (step A2) and then transfer and write operations (step A3) are performed in an order named.

More specifically, as shown in Fig. 3B, frame display data is read out from a predetermined memory address and transferred to a display memory (step B1). Cursor data is then generated and transferred to the display memory (step B2). A key input is checked (step B3). The key input is transferred to a buffer and the display memory (step B4). A cursor process is performed to move the cursor to the next write position (step B5). A process for determining whether a digit position count exceeds a predetermined count (step B7) is required.

In this program generation, predetermined steps (skeleton 20) are surrounded by a dotted line in FIG. 3B. Other steps constitute a variable portion which can be changed in accordance with the needs of the user. If a frame shown in FIG. 3C is requested as a data input frame by the user, a frame pattern must be designated in step B1. By this designation, a data write area assured in the display memory is set (step B4). A parameter for a comparison process for determining whether the digit position count exceeds the digit position count per line (step B7) is also determined.

In the cursor generating process (step B2), the cursor data may be fixed if there is a program for searching a non-recorded portion in a frame to transfer the cursor data to a position having a minimum digit position count and a minimum line count. Even in a one-line confirmation button process (step S6), coding may be fixed such that a line return process is initiated when depression of a confirmation button is confirmed;

otherwise, the flow advances to step B7.

When this skeleton 20 is designated, a prompt for requesting data inputs for designing a frame is displayed on the workstation 200. This request includes data of a digit position count, a lateral block count, and a vertical line count. When all the requested data are filled, the program generating section 100 displays a message representing the end of input, and the coding process is performed.

The frame, data input states, input data request messages, and a keyword to be input, and the like are displayed on the screen. For this reason, the user can select the keyword to input numerical values and characters, thereby facilitating program creation.

The above example is very simple and represents program generation of only a main part. However, in actual coding, more minute steps are required.

The number of steps of each skeleton itself is very large. In the above system, a plurality of skeletons are stored in the skeleton file 300.

In the above description, FIG. 2 shows all the steps 11 of the given program X, and a portion surrounded by the chain line in FIG. 2 is defined as the skeleton, as described above. However, in storage of this skeleton data in the skeleton file, the assurance of the area for the undefined portions 13 wastes the storage capacity. That is, the capacity of the skeleton file 300 is wasted.

As shown in FIG. 4, in this system, flags representing the undefined portions 13 are inserted in correspondence with these undefined portions to reduce the capacity of the undefined portions 13. For this reason, even if a skeleton has a large number of variable portions and a very large undefined portion, it can be compressed and stored in a small capacity in the skeleton file 300.

When the skeleton is read out from the program

generating section 100 to fill the flag portions with external inputs during program creation, the resultant program has a prescribed length.

5 When the skeleton is loaded in the work memory, the program generating section 100 has a function of searching the flags in an order of program steps. When each flag is detected, data to be input is requested to the workstation 200 in accordance with this flag number. The program generating section 100 also has a function
10 of determining whether a jump instruction is present in the skeleton or added steps to manage and correct the address (step number) of the jump instruction. When the undefined portions of the skeleton are filled with the external data, the jump address (step) changes, and this
15 change is corrected finally by the function of the program generating section 100.

FIG. 5 is a flow chart exemplifying this operation.

The flow starts in step C1 to load the readout skeleton in the program generating section (builder) 100
20 from the skeleton file 300. The program generating section 100 increments the address to confirm program data upon sequential reading the skeleton steps (steps C2 and C3). The program generating section 100 determines whether program end data is present (step C4). If NO in
25 step C4, the program generating section 100 determines whether a flag is present (step C5). If a flag is found, program data corresponding to this flag is read, and data to be input is requested to the workstation 200 (step C6). The request form is of a worksheet type.
30 This is a fixed format corresponding to a request content. The user can fill only requested variable items (numerical values and symbols) in the predetermined portions of the worksheet displayed on the workstation. If all the request items are satisfied, no omissions occur
35 in conditions and the like. That is, coding does not vary depending on programmers, but can be uniformly performed.

When the data input is completed, and a key input representing the end of input is detected (step C7), the step addresses of the subsequent skeletons are corrected in accordance with the decrease/increase in the number of steps specified by the user in the next step (step C8). The subsequent step addresses of the skeleton may be increased by the variable items, or an unnecessary routine may be partially deleted or decreased. The unnecessary routine can be selectively deleted in accordance with the requested items stored at the program data addresses of the flags.

A step address management map for sequentially managing the original skeleton step (address) and all the steps (addresses) of the instruction words upon partial correction of the skeleton is created in step C8. In this manner, when the flag portions are filled with the external data, and program end data is detected, the flow advances to step C9 to return to the initial step of the created program. The address is then incremented to advance the step to determine whether a jump instruction is present in this position (steps C10 and C12). If a jump instruction is detected, the jump destination address represented by this instruction must be corrected. The step address management map which has already managed the addresses is accessed to detect a specific address to which the jump destination address is to be changed, thereby correcting the jump destination address to a new jump destination address (step C13). In this manner, the jump destination addresses are sequentially corrected. When the program end data is detected, this data input process is ended (steps C14 and C15).

The system described above compresses and stores the skeleton steps to reduce the capacity for storing the skeleton. When variable items and component items are added to the flag portions, the jump destination addresses of the jump instructions are changed, so that

the system has a jump destination address correction function.

FIGS. 6 and 7 show ordinary words obtained by partially converting a skeleton used to generate, e.g., a client master file program. The steps (addresses) of the skeleton are located at the left end, and variable items and components are input to the flag positions (undefined areas).

FIG. 8 shows ordinary words obtained by partially converting a program obtained such that variable items and components are inserted in the flag positions of the skeleton and the resultant structure is coded. Steps 1.00 to 29.00 and 33.00 to 35.00 at the left end are steps automatically generated using the variable items. Steps 30.00 to 32.00 and 40.00 to 43.00 are steps automatically generated using the components. Steps 36.00 to 39.00 and 44.00 to 60.00 are steps originally present in the skeleton.

Various skeletons are illustrated from FIG. 9. FIGS. 9 and 10 show part of a batch update skeleton. FIGS. 11 and 12 also show part of a batch update skeleton. FIGS. 13 and 14 also show part of another skeleton. Each pair of drawings in FIGS. 15 to 62 show part of one skeleton.

As has been described above, according to the present invention, a program generating process can be a plant-like process. An accurate, error-free program can be uniformly generated in accordance with an industrial technique. In addition, the design steps are simple jobs to shorten the program generation time.



Claims:

1. An automatic program generating apparatus characterized by comprising:

storage means (300) for storing a plurality of skeletons each having a format constituted by instruction data of fixed process items for regular job contents;

means (100) for reading out a skeleton of a designated format;

means (100) for requesting to input predetermined instruction words in undefined instruction areas of the readout skeleton in a predetermined order; and

program coding means (100, 500) for determining that necessary instruction data are input to all the undefined instruction areas of the readout skeleton and coding a resultant structure to generate a program.

2. An apparatus according to claim 1, characterized in that the plurality of skeleton are stored in said storage means such that addresses of portions corresponding to the undefined areas are compacted to effectively utilize a capacity of said storage means.

3. An apparatus according to claim 1, characterized by further comprising means for correcting a jump destination address of a jump instruction when the jump instruction is present in the fixed process items in a state wherein the instruction words are inserted in the undefined areas of the skeleton.

4. An apparatus according to claim 1, characterized in that said program coding means further comprises means for reading coded program data upon program coding, inversely converting the read coded program data, and automatically generating a program document representing a structure and procedures of the program.

5. An automatic program generating apparatus characterized by comprising:





storage means (300) for storing a plurality of skeletons, each of the plurality of skeletons serving as a framework for generating a program and having a step area for instructions whose data process procedures are predetermined and a step area for undefined instructions;

display means (200) for displaying characters representing job contents corresponding to the plurality of skeletons stored in said storage means;

selecting means (200) for inputting an identification code of a job content displayed on said display means from a keyboard;

readout means (200, 100) for reading out any one of the plurality of skeletons in response to the identification code input from said selecting means, the readout skeleton waiting in a program generating section;

input request means (100) for displaying and requesting, through said display means, instruction data which is to be set in the step area for the undefined instructions and included in the skeleton waiting in said program generating section; and

automatic coding means (100, 500) for converting a program into a machine language after the step area for the undefined instructions of the waiting skeleton is filled with the instruction data.

6. An apparatus according to claim 5, characterized in that said input request means comprises:

means (C1 - C5) for sequentially checking program steps of the waiting skeleton to detect whether a flag is present in the program steps;

means (C6) for causing said display means to display an instruction data input request when the flag is detected; and

means (C8) for correcting step addresses of the waiting skeleton in accordance with a change in a number of steps when the instruction data is input in response





to the input request.

7. An apparatus according to claim 6, characterized by further comprising:

5 means (C8) for creating an address map representing a correspondence between the step addresses of the waiting skeleton before correction and those after the correction when the step addresses of the waiting skeleton are corrected;

10 means (C4) for detecting a program end instruction of the waiting skeleton;

means (C12) for detecting whether a jump instruction is present in the steps of the waiting skeleton after the program end instruction is detected; and

15 means (C13) for correcting a jump destination address designated by the jump instruction with reference to the address map when the jump instruction is present.

