프로그램 메모리 구조

|  |
| --- |
| Dos 영역 |
| : |
| 코드 세그먼트(기계어 변역) |
| 초기화 데이터 영역 특정한 값 설정 ex)주소 |
| 비초기화 데이터 영역 실제 사용될 때 사용되는 공간 |
| 근거리 힙 |
| 스택 |
| 원거리 힙 |
| : |

SBCS: 아스키코드, 영문과 숫자만 사용가능-> 1바이트/MBCS: 멀티바이트, 한글은 2바이트, 영문과 숫자 1바이트  
WBCS: 유니코드 각각 2바이트

String 클래스 -> <string>: 문자열 처리과정에서 발생하는 메모리 공간에 대한 관리를 자동으로 수행

|  |  |
| --- | --- |
| 생성자 | 설명 |
| string(const char \*s) | char \*s 포인터로 전달된 문자열을 이용해서 string을 초기화한다. |
| string(size\_type n, char c) | n개의 용소를 가진 string 개체를 생성하고, c문자로 채운다. |
| string() | 크기가 0인 기본 string 개체를 생성한다. |
| string(const char \*s, size\_type n) | char \*s 포인터가 지시된 문자열을 이용해서 최대 n번째 문자까지만 저장되는 string객제를 생성한다. |

String의 크기와 용량 조절

string에 저장한 실제 문자와 총 개수(크기)->size(), length()

메모리 재할당 과정 없이 입력될 수 있는 크기(용량)->capacity()

시스템에서 할당 받을 수 있는 최대 문자의 개수를 알아냄->max\_size()

불필요한 메모리 관리를 생략하기 위한 용량 조절->reserve()

String 객체에 문자가 없는 경우 true를 반환함

문자열의 대입 및 할당 및 검색

변수1 = 변수2;

할당받는 변수.assign(소스);

할당받는 변수.assign(소스, 첫번째 문자열 위치, 소스 글자수);

find(변수): 문자열 검색 /rfind(변수): 끝에서부터 검색 시작-> find와 동일함

|  |  |  |  |
| --- | --- | --- | --- |
| 문자열 추가 | 문자열 교환 | 문자열 삭제 | 문자열 치환 |
| 변수 += 문자열 | 변수.swap(변수) | 변수.erase(변수시작위치,  변수의 글자수) | 변수.replace(변수시작위치, 변수 글자수,  교체할 문자) |
| 문자열 복사 | 부분 문자열 얻기 | 문자열 삽입 | |
| 변수.copy(소스,  시작위치, 글자수) | 변수.substr(시작위치)  변수.substr(시작위치, 글자수) | 변수.insert(변수 시작위치, 소스) 변수.insert(변수 시작위치, 소스, 소스 시작위치, 글자수) | |

문자열과 문자배열 차이점  
‘\0’ 자동삽입차이, 큰따옴표와 작은따옴표 차이

문자열 조작 함수  
strcpy(변수, 소스): 문자열 복사-> \0문자 포함  
strncpy(변수, 소스): 문자열의 글자 수 만큼 복사->\0문자 불포함

WBSC를 위한 두가지  
1. Char->wchar\_t/2. “문자열”대신 L“문자열”

|  |  |  |  |
| --- | --- | --- | --- |
| SBCS함수 | WBCS함수 | SBCS함수 | WBCS함수 |
| strcpy | wcscpy | strlen | wcslen |
| strncpy | wcsncpy | prinf | wpirnf |
| strcat | wcscat | scanf | wscanf |
| strncat | wcsncat | fgets | fgetws |
| strcmp | wcscmp | fputs | fputws |
| strncmp | wcsncmp |  |  |

32비트와 64비트 비교

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 비트 | 모델 | char | short | int | long | 포인터 |
| 32비트 (=4바이트) | Window | 1 | 2 | 4 | 4 | 4 |
| 64비트 (=8바이트) | Window (LLP64) | 1 | 2 | 4 | 4 | 8 |
| UNIX  (LP64) | 1 | 2 | 4 | 8 | 8 |

☞32비트 x86/64비트 x64

파일에 데이터를 저장하기 위한 과정  
1. #include <fstream> ->입출력 라이브러리

2. 출력 스트림을 관리하기 위한 객체 선언 ex) ofsream fout;

3. 선언된 출력 스트림 객체에 특정 파일을 지정해 연다. ex) fout.open(“insa.dat”);

4. 선언된 객체에 출력 전용 연산자(<<)나 맴버함수를 통해서 출력  
ex) fout << male1.name << " " << male1.age << " " << male1.address << endl;

5. 열었던 파일을 닫는다. ex) fout.close();

파일 입력을 위한 개념  
1. 입력 스트림을 관리하기 위한 객체를 선언 ex)ifstream fin

2. 입력 스트림을 관리하기 위한 객체를 선언

3. 선언된 입력 스트림 객체에 특정 파일을 지정해 연다. ex) fin.open(“insa.dat”);

4. 파일에서 데이터를 일기 위해 파일을 연 경우에는 해당 파일이 반드시 존재해야 한다. 파일 입력 스트림 객체로 open 맴버함수를 호출한 후에 파일 열기에 성공했는지 알아보기 위해서 다음 표에 있는 함수가 사용된다.

|  |  |
| --- | --- |
| 함수 | 설명 |
| eof() | 파일의 끝을 알리는 EOF를 만나면 참을 반환한다. -> 정상적인 상황 |
| bad() | 올바르지 못한 입출력을 시도하려 할 때 참을 반환한다. -> io에 심각한 상황 |
| fail() | 올바르지 못한 입출력 일어났을 때 참을 반환한다. -> 충분히 고칠 수 있다. |
| good() | 파일을 성공적으로 열거나 위의 세 함수가 거짓이면 참이다. |

5. 선언된 객체에 입력 전용 연산자(“>>”)나 맴버함수를 통해야 입력받는다.  
ex) fin >> male1.name >> " " >> male1.age >> " " >> male1.address;

6. 열었던 파일을 닫는다.  
fin.close();